ORACLE

CLINTRIAL™

Reference Guide

release 4.7.1

Part Number: E27576-01

Contents

Preface

Overview

Clintrial™ 4 software (hereafter referred to as Clintrial software) is a comprehensive clinical research system for the collection, management, and review of clinical trials data. Clintrial software is designed for use by companies that must both:

- Collect clinical data to meet regulatory requirements for conducting clinical trials.
- Analyze data that is collected during those clinical trials.

Clintrial software enables companies to unify all of their clinical data collection and management, regardless of source or phase of development (pre- or post-market).

About this book

This book is written for all Clintrial software users. It explains Clintrial software concepts and describes the tasks you can perform with Clintrial software. Other chapters cover product installation, and setup of the Sample Studies.

About the Clintrial software documentation

The Clintrial software documentation includes books that contain conceptual information. The Clintrial software Help contains procedures for the tasks that you perform with the Clintrial software.

The Clintrial software documentation assumes that you know how to perform basic tasks on your computer.

What are the Clintrial software books?

The Clintrial 4.7 documentation includes the documents in the following table. All documentation is available from the Phase Forward Download Center.

Title:	**Content:**
Release Notes	The *Release Notes* document describes enhancements introduced and problems fixed in the current release, upgrade considerations, release history, and other late-breaking information.
Known Issues	The *Known Issues* document provides detailed information about the known issues in this release, along with workarounds, if available. **Note:** The most current list of known issues is available on the Phase Forward Extranet. To sign in to the Extranet, go to https://extranet.phaseforward.com and click **Customer Login**. Enter your email address and password, and navigate to the **Known Issues** section. Select a product, and then enter your search criteria.
Getting Started	The *Getting Started* guide: • Provides a summary of each Clintrial module, a description of the relationships between modules, and descriptions of key concepts. • Describes how to install, upgrade, and de-install the Clintrial software. • Describes how to configure the Clintrial application. • Provides information and procedures for customizing the Windows Registry. • Explains how to use the Medika Sample Studies.
Admin and Design	The *Admin and Design* document describes how to use: • The Admin module to work with user accounts, access rights, parameters, and system administration tools. • The Design module to set up and maintain Clintrial application objects, such as protocols, panels, and study books.
Secure Configuration Guide	The *Secure Configuration Guide* provides an overview of the security features provided with the Clintrial application including details about the general principles of application security, as well as how to install, configure, and use the Clintrial application securely.

Title:	Content:
Reference Guide	The *Reference Guide* provides: • Definitions of the Oracle database tables that store Clintrial metadata and clinical data. • Descriptions of the use of PL/SQL for Clintrial-specific procedures. • Explanations of data types and naming conventions. • Information on using SQL, setting up custom menus, and running batch jobs. • A glossary of terms.
Manage, Classify, and Lab Loader	The *Manage, Classify, and Lab Loader* document describes how to use: • The Manage module to perform data management tasks such as coding (including integration with Central Coding), global modification, validation, auditing, and batch loading of clinical data. • The Classify module to track, review and solve for values that fail automatic coding; to audit the contents of a coding thesaurus protocol; and to build and test effective thesaurus algorithms. • The Lab Loader module to batch load laboratory data and to set up Lab Loader objects.
Enter, Resolve, and Retrieve	The *Enter, Resolve, and Retrieve* document describes how to use: • The Enter module to enroll subjects, enter and edit data, verify data, and work with reports. • The Resolve module to identify, track, and report data discrepancies, as well as how to customize the Resolve module, including writing rules that reference data items. • The Retrieve module to extract clinical data from the database and work with query results.
Multisite	The *Multisite* document describes: • How to distribute codelists and protocols. • How to set up a replication environment. • How other Clintrial modules work differently in a Multisite environment.
Quick Reference Card for Enter	The *Quick Reference for Enter* lists Enter module menu commands and shortcut keys.

Conventions

The following conventions are used in the Clintrial software books:

Convention:	**Description:**
Italics	Italics are used to indicate the following: • New terms • Titles of books • Variable names in code examples or file names
Ctrl + c	Key combinations where you press the first key and hold it down while you press the second key. For example, to copy selected text to the clipboard, you press the **Ctrl** key and hold it down while pressing the **c** key.
bold	Menu names, command names, dialog box buttons, and key names appear in bold type. Additionally, the text you enter in fields during procedures appears in bold type.
COMMENT IS NULL	Examples of programming code (such as PL/SQL) or SQL commands are emphasized with a different font.
	This caution symbol advises users that failure to take or avoid a specified action could result in significant data problems.

Medika Sample Studies

The Clintrial software provides three sample studies that you can optionally install and use as a learning aid.

For information about installing and using the sample study, see the *Clintrial Getting Started* guide, Chapter 7.

Clintrial 4.7 compatibility with other Oracle Health Sciences products

The *Products Compatibility Matrix*, which identifies Clintrial compatibility with other Oracle Health Sciences products, can be downloaded from https://extranet.phaseforward.com.

To sign in, click **Customer Login**. Enter your email address and password, and navigate to the **Bulletins** section.

If you need assistance

If you are an Oracle customer with a maintenance agreement, you can contact the Global Support Center for assistance with product issues.

Your maintenance agreement indicates the type of support you are eligible to receive and describes how to contact Oracle. Additionally, the Oracle website lists the toll-free support number for your product, location, and support level:

http://www.oracle.com/support/

In the event that our toll-free telephone service is interrupted, please use either of the following methods to contact the Global Support Center:

- email

  saasclinicalsupport_ww@oracle.com
- telephone

In the US:	1-800-633-0925
Outside of the US:	+44 (0) 207 131 2801

Oracle also provides assistance with User Management, Site Assessment, and Provisioning. Please refer to your Master Services Agreement and individual Statement of Work to determine if you are eligible to use these services.

Part I: Database Structures

1 Introducing Database Structures

How to use this part of the guide

Part I provides the following information about each Oracle table that is in a protocol account, or in certain system accounts, such as CTS, CTSCODES or CTSDD:

- A description of the type of information stored in the table.
- A description of rows in the table.
- A description of columns in the table, including column names, data types, and meanings.
- A description of the indexes for each table, including index names, whether the indexes are unique, and the names of indexed columns.

The meaning of certain columns is shown as obsolete or reserved. If obsolete, then the column has been retained in the tables for compatibility with previous Clintrial software releases; however, the column is no longer used. If reserved, then the column is not currently used, but it may be needed in future Clintrial software releases.

Caution*:* Access to these tables outside of Clintrial software activities should be read-only. The presentation of this material should *not* be construed as explicit or implicit permission to alter the contents of the tables outside of Clintrial software activities. We cannot guarantee that the Clintrial software will operate correctly if you modify the contents of these tables. Also note that the format and content of these tables is subject to change in subsequent releases of the Clintrial software.

Types of accounts

Protocol accounts

A *protocol account*, or protocol, is an Oracle account that stores information that is specific to a protocol, including the following:

- Clinical data for a particular clinical study or group of studies.
- Error log records, flags, and notes associated with the clinical data.
- Views of clinical data.

See the *Design* section of *Admin and Design* for information about protocol accounts.

System accounts

A Clintrial software system account is an Oracle account that stores database-wide information. The *Design* section of *Admin and Design* describes the system accounts. This guide describes tables in the following system accounts:

- CTS account — The CTS account stores information about: protocol accounts; flags and notes; user accounts and privileges; Clintrial software activities, messages and parameters; and Oracle parameters.
- CTSCODES account — The CTSCODES account stores information about codelists.
- CTSDD account — The CTSDD account stores information about protocol-specific objects, such as panels and items.
- CTCLASSIFY account — The CTCLASSIFY account is created automatically by the installation of Classify. This account stores information about Classify-specific objects, such as omissions.
- CTSRM account — The CTSRM account is created automatically when you install the Distribution (CTC) server component of Multisite. This account stores information that is used to manage distribution across multiple sites.
- CTSRP account — The CTSRP account is created automatically when you install the Replication (CTX) server component of Multisite. This account stores information that is used to manage replication across multiple sites.

2 Protocol Account

*panel-name*_UPDATE

For each installed panel for which database tables exist, there is an update table. The *update table* stores clinical data that has been entered but is not yet merged (that is, moved to the data table). The full name of the update table is *protocol-name*.*panel-name*_UPDATE.

The Columns section following describes the system items that comprise the first eight columns of the update table, and are used internally by the Clintrial software to identify records. Other columns in the update table are determined by items defined for the CONTEXT panel and for the specific panel.

Rows

For information about how records are stored in the update table for different panel types, see the *Design* section of *Admin and Design*.

Columns

Column name:	Data type:	Description:
MERGE-_DATETIME	DATE	Date and time at which the record was created, or was last modified in the update table. (Modifications include changes to any items, including other system items, in the record.)
STATUS	NUMBER(2)	Numeric code indicating the status of the record: 3 — Batch-loaded and not yet screened. 2 — Entered interactively and not yet verified. 1 — Passed verification or screening. 0 — Passed validation. –1 — Failed validation or merge. –2 — Failed verification. –3 — Failed screening.

Column name:	Data type:	Description:
ENTRY_ID	VARCHAR2(20)	User account that entered or last modified the record. (For batch-loaded records, this value is initially CTS$LOAD.)
ENTRY- _DATETIME	DATE	Date and time at which the record was created.
CT_RECID	VARCHAR2(40)	Unique identifier that is automatically assigned to the record. The CT_RECI, for example, 1,SMITH.LFYKQ[c.001, consists of four parts. The first part (1 in the preceding example) identifies the database in which the record was created. If Multisite is not installed, this value is always 1. The second part (SMITH in the preceding example) identifies the user account that created the record. For batch-loaded records, this part includes the characters SQLLOAD. The third part (LFYKQ[c in the preceding example) identifies the observation, and is the same for all records in an observation. The fourth part (001 in the preceding example) distinguishes records that are part of the same observation. The fourth part of the CT_RECID is normally three characters long if there are less than 999 records in an observation. For a non-repeating record, this value is 001. For repeating records, if a new record is inserted between existing repeating records, then the new record will have a CT_RECID the same as the CT_RECID of the preceding record, but with a single digit appended at the end. For example, if there are two records, one with .002 and another with .003, at the end of the CT_RECID and a record is inserted between the two, the new record will have a CT_RECID with .0021 at the end.
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.

Column name:	Data type:	Description:
SUBJECT_ID	NUMBER(15)	Unique identifier that is automatically assigned to the subject item. The subject item is a subject-related context item that the designer designates as the unique key for identifying data by subject. For example, suppose that the subject-related context items are SUB_ID and INV_ID, and that the designer designates SUB_ID as the subject item. For each unique SUB_ID value, there will be a unique SUBJECT_ID value.
CTS$REASON	VARCHAR2(2000)	Reason that the record was changed or deleted.

Indexes

Index name:	Unique?:	Columns indexed:
*panel-name*_UPK (all panel types)	Yes	CT_RECID
*panel-name*_UPIDX (panel types 1 – 5)	No	SUBJECT_ID, *block_item* (the context item defined as the block key item)

*panel-name*_DATA

For each installed panel for which database tables exist, there is a data table. The *data table* stores clinical data that has been merged, that is, moved from the update table to the data table. The full name of the data table is *protocol-name.panel-name*_DATA.

The following Columns section describes the system items that comprise the first eight columns of the data table, and are used internally by the Clintrial software to identify records. Other columns in the data table are determined by items defined for the CONTEXT panel and for the specific panel.

Rows

For information about how records are stored in the data table for different panel types, see the *Design* section of *Admin and Design*.

Columns

Column name:	Data type:	Description:
MERGE- _DATETIME	DATE	Date and time at which the record was merged, or was last modified in the data table. (Modifications include changes to any items, including other system items, in the record.) If the record was entered directly into the data table (as for Type 5 panels), this is the date and time the record was created in the data table.
STATUS	NUMBER(2)	Numeric code indicating the status of the record. For a record in the data table, this status is always 0, indicating that the record passed validation.
ENTRY_ID	VARCHAR2(20)	User account of the user who entered or last modified the record.
ENTRY- _DATETIME	DATE	Date and time at which the record was created in the update table. If the record was entered directly into the data table (as for Type 5 panels), this is the date and time the record was created in the data table.
CT_RECID	VARCHAR2(40)	Same value as that of the record when it was moved from the update table to the data table.
DB_ID	NUMBER(5)	Same value as that of the record when it was moved from the update table to the data table.
SUBJECT_ID	NUMBER(15)	Same value as that of the record when it was moved from the update table to the data table.

Column name:	Data type:	Description:
CTS$REASON	VARCHAR2(2000)	Reason that the record was modified or deleted.

Indexes

Index name:	Unique?:	Columns indexed*:
*panel-name*_DPK (all panel types)	Yes	CT_RECID
*panel-name*_DBIDX (Type 1 panel)	Yes	SUBJECT_ID
*panel-name*_DBIDX (Type 2 panel)	No	SUBJECT_ID
*panel-name*_DBIDX (Type 3 panel)	Yes	SUBJECT_ID, block key item, and block repeat key item (if defined)
*panel-name*_DBIDX (Type 4 panel)	No	SUBJECT_ID, block key item, and block repeat key item (if defined)
*panel-name*_DBIDX (Type 5 panel)	Yes	SUBJECT_ID
*panel-name*_SBIDX (Type 5 panel)	Yes	Subject item (the context item defined as the subject item)

* If there are user-defined panel keys for a Type 0, 2, or 4 panel, then the index includes those keys. For a Type 2 or 4 panel with defined panel keys, the index also includes the page item and page repeat key item.

*panel-name*_AUDIT

For each installed panel for which database tables exist, there is an audit table. The *audit table* stores copies of records that were modified or deleted while they were in the update table or data table and auditing was in effect. The full name of the audit table is *protocol-name*.*panel-name*_AUDIT.

The following Columns section describes the system items that comprise the first eight columns of the audit table, and are used internally by the Clintrial software to identify records. Other columns in the audit table are determined by items defined for the CONTEXT panel and for the specific panel.

Rows

For information about how records are stored in the audit table for different panel types, see the *Design* section of *Admin and Design*.

Columns

Column name:	Data type:	Description:
MERGE- _DATETIME	DATE	If the record was modified or deleted, this value is the same as the MERGE_DATETIME of the record that was modified or deleted. ***Note*:** For a deleted record, a second copy of the deleted record is placed in the audit table. For the second copy, the MERGE_DATETIME is the date of the deletion.

Column name:	Data type:	Description:
STATUS	NUMBER(2)	Numeric code indicating the status of the record. -60 — Was modified when it had the status 2. -61 — Was deleted when it had the status 2. -50 — Was modified when it had the status –2. -51 — Was deleted when it had the status –2. -40 — Was modified when it had the status 1. -41 — Was deleted when it had the status 1. -30 — Was modified when it had the status –1. -31 — Was deleted when it had the status –1. -20 — Was modified when it had the status 0 and was in the update table. -21 — Was deleted when it had the status 0 and was in the update table. -10 — Was modified when it had the status 0 and was in the data table. -11 — Was deleted when it had the status 0 and was in the data table.
ENTRY_ID	VARCHAR2(20)	For a record that was modified or deleted, the ENTRY_ID is that of the record when it was modified or deleted. ***Note***: For a deleted record, a second copy of the deleted record is placed in the audit table. For the second copy, the ENTRY_ID is the user account of the user deleting the record.
ENTRY- _DATETIME	DATE	Date and time at which the record was placed in the audit table. ***Note***: For a deleted record, a second copy of the deleted record is placed in the audit table. For the second copy, the ENTRY_DATETIME is the same as the ENTRY_DATETIME of the record that was deleted.

Column name:	Data type:	Description:
CT_RECID	VARCHAR2(40)	Same value as that of the record when it was modified or deleted in the update table or data table.
DB_ID	NUMBER(5)	Same value as that of the record when it was modified or deleted in the update table or data table.
SUBJECT_ID	NUMBER(15)	Same value as that of the record when it was modified or deleted in the update table or data table.
CTS$REASON	VARCHAR2(2000)	Reason that the record was modified or deleted.

Indexes

Index name:	Unique?:	Columns indexed:
*panel-name*_APK (panel types 0 – 5)	Yes	CT_RECID, MERGE_DATETIME
*panel-nam*e_APIDX (panel types 1 – 5)	No	SUBJECT_ID

ERRORLOG

The *protocol-name*.ERRORLOG table stores information about errors that occur during screening, validation, merging, and global change or deletion.

Rows

One per error.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol in which the error occurred.
PANEL	VARCHAR2(20)	Name of the panel containing the data for which the error occurred.
CT_RECID	VARCHAR2(40)	Value of the CT_RECID system item for the record for which the error occurred.
SUBJECT_ID	NUMBER(15)	Value of the SUBJECT_ID system item for the record for which the error occurred.
BLOCK_ITEM	VARCHAR2(240)	Value of the block item for the record for which the error occurred.
PAGE_ITEM	VARCHAR2(240)	Value of the page item for the record for which the error occurred.
BLOCK_REPEAT-_ITEM	VARCHAR2(40)	Value of the block repeat key item for the record for which the error occurred.
PAGE_REPEAT-_ITEM	VARCHAR2(40)	Value of the page repeat key item for the record for which the error occurred.
ORCTABLE	VARCHAR2(80)	UPDATE — The error occurred for a record in the update table. DATA — The error occurred for a record in the data table.
ERRDT	DATE	Date and time at which the error occurred.
ERRTYPE	VARCHAR2(20)	Clintrial software activity during which the error occurred: GLOBAL CHG — Global change GLOBAL DEL — Global deletion MERGE — Merge SCREEN — Screening VALIDATE — Validation

Column name:	Data type:	Description:
REMARKS	VARCHAR2(240)	Message text associated with the error.
ERRACT	VARCHAR2(20)	If ERRTYPE is VALIDATE: • REPORT — The record failed the rule, and the rule action is REPORT. • REJECT — The record failed the rule and the rule action is REJECT. If ERRTYPE is SCREEN: • REPORT — The record failed data checks but the checks were overridden. • REJECT — The record failed data checks and the checks were not overridden. For all other error types, this value is REJECT if the activity failed.
RULE_NAME	VARCHAR2(20)	If ERRTYPE is VALIDATE, the name of the rule that resulted in an error.
REC_MODDATE	DATE	Date and time at which the record for which the error occurred was created or last modified.

Index

Index name:	Unique?:	Columns indexed:
ERRORLOG_INDX	No	PANEL, ERRTYPE

SUBJECT_BLOCK

The *protocol-name*.SUBJECT_BLOCK table stores information about blocks for which data exists for a subject.

Rows

One per block per subject.

Columns

Column name:	Data type:	Description:
SUBJECT_ID	NUMBER(15)	Identifier of a subject.
BLOCK_KEY	VARCHAR2(40)	Block key value for the block.
BLOCK_REPEAT-_KEY	VARCHAR2(40)	Block repeat key value for the block.
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.

Index

Index name:	Unique?:	Columns indexed:
SUBJ_BLOCK_PK	Yes	SUBJECT_ID, BLOCK_KEY, BLOCK_REPEAT_KEY

SUBJECT_PAGE

The *protocol-name*.SUBJECT_PAGE table stores information about study pages for which data exists for a subject.

Rows

One per study page per subject.

Columns

Column name:	Data type:	Description:
SUBJECT_ID	NUMBER(15)	Identifier of a subject.
BLOCK_KEY	VARCHAR2(40)	Block key value for the page.
BLOCK_REPEAT-_KEY	VARCHAR2(40)	Block repeat key value for the page.
PAGE_KEY	VARCHAR2(40)	Page key value for the page.
PAGE_REPEAT-_KEY	VARCHAR2(40)	Page repeat key value for the page.
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.
STATUS_0	NUMBER(15)	Holds the number of records on the current page with status value = 0.
STATUS_1	NUMBER(15)	Holds the number of records on the current page with status value = 1.
STATUS_2	NUMBER(15)	Holds the number of records on the current page with status value = 2.
STATUS_N1	NUMBER(15)	Holds the number of records on the current page with status value = -1.
STATUS_N2	NUMBER(15)	Holds the number of records on the current page with status value = -2.
DISCREP_CNT_S	NUMBER(15)	Holds the number of discrepancies on the current page with discrepancy state = Sent (S).
DISCREP_CNT-_ RTS	NUMBER(15)	Holds the number of discrepancies on the current page with discrepancy state = Ready to Send (RTS).

Column name:	Data type:	Description:
DISCREP_CNT	NUMBER(15)	Holds the number of discrepancies on the current page with all other open discrepancy states.
PAGE_STATE	NUMBER(1)	Indicates the state of a page in the Connect 1.1 workflow: 0 — Connect 1.1 is not installed. 2 — Draft. 3 — Submitted. 4 — Signed.

Indexes

Index name:	Unique?:	Columns indexed:
SUBJ_PAGE_PK	Yes	SUBJECT_ID, BLOCK_KEY, BLOCK_REPEAT_KEY, PAGE_KEY, PAGE_REPEAT_KEY

TAGS

The *protocol-name*.TAGS table stores information about tags (that is, flags and notes) that are attached to clinical data.

Rows

One per flag or note that is attached to clinical data.

Columns

Column name:	Data type:	Description:
TAG_RECID	NUMBER(15)	Unique identifier that is assigned automatically to the flag or note.
CT_RECID	VARCHAR2(40)	Value of the CT_RECID system item for the record to which the tag is attached, or for the record containing the item to which the flag or note is attached.
TAGID	NUMBER(15)	Unique identifier of the type of flag or note.
AGGREGATION	VARCHAR2(1)	Level of data to which the flag or note is attached: O — Observation R — Record I — Item
PANEL	VARCHAR2(20)	Name of the panel containing the data to which the flag or note is attached.
DBTABLE	VARCHAR2(1)	Type of table containing the data to which the flag or note is attached: UPDATE — Update table DATA — Data table
ENTRY_ID	VARCHAR2(20)	User account that attached or last modified the flag or note.
ENTRY_DT	DATE	Date and time at which the flag or note was attached.
MODIFY_DT	DATE	Date and time at which the flag or note was last modified.
ITEMNAME	VARCHAR2(20)	If AGGREGATION is I, the name of the item to which the flag or note is attached.
BATCH_ID	VARCHAR2(7)	Obsolete.

Column name:	Data type:	Description:
SUBJECT_ID	NUMBER(15)	Value of the SUBJECT_ID system item for the record to which the flag or note is attached.
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.
NOTE	VARCHAR2(2000)	Flag comment or note text.

Indexes

Index name:	Unique?:	Columns indexed:
TAGSIDX2	No	CT_RECID
TAGSIDX3	Yes	PANEL, CT_RECID, TAGID, AGGREGATION, ITEMNAME
TAGS_PK	Yes	TAG_RECID

TAGS_AUDIT

The *protocol-name*.TAGS_AUDIT table stores information about modified or deleted flags or notes.

Rows

One per modification or deletion of a flag or note.

Columns

Column name:	Data type:	Description:
TAG_RECID	NUMBER(15)	Same value as that of the flag or note when it was modified or deleted.
CT_RECID	VARCHAR2(40)	Same value as that of the flag or note when it was modified or deleted.
TAGID	NUMBER(15)	Same value as that of the flag or note when it was modified or deleted.
AGGREGATION	VARCHAR2(1)	Same value as that of the flag or note when it was modified or deleted.
PANEL	VARCHAR2(20)	Same value as that of the flag or note when it was modified or deleted.
DBTABLE	VARCHAR2(1)	Same value as that of the flag or note when it was modified or deleted.
ENTRY_ID	VARCHAR2(20)	User account that modified or deleted the flag or note.
ENTRY_DT	DATE	Same value as that of the flag or note when it was modified or deleted.
MODIFY_DT	DATE	Date and time at which the flag or note was modified or deleted.
ITEMNAME	VARCHAR2(20)	Same value as that of the flag or note when it was modified or deleted.
BATCH_ID	VARCHAR2(7)	Obsolete.
SUBJECT_ID	NUMBER(15)	Same value as that of the flag or note when it was modified or deleted.
DB_ID	NUMBER(5)	Same value as that of the flag or note when it was modified or deleted.
NOTE	VARCHAR2(2000)	Same value as that of the flag or note when it was modified or deleted.

Index

Index name:	Unique?:	Columns indexed:
TAGS_AUDIT_PK	Yes	TAG_RECID, MODIFY_DT

VCT_ERRORITEM_UPDATE

When you set up a protocol for Resolve, the VCT_ERRORITEM panel is copied from the CTRESOLVEREF protocol to that protocol.

For a description of the VCT_ERRORITEM panel, see Chapter 10.

VCT_ERRORSTATUS_UPDATE

When you set up a protocol for Resolve, the VCT_ERRORSTATUS panel is copied from the CTRESOLVEREF protocol to the protocol.

For a description of the VCT_ERRORSTATUS panel, see Chapter 10.

BLOCK_REPEAT_ITEM

3 CTS Account

ACCESS_RIGHT

The CTS.ACCESS_RIGHT table stores information about protocol and non-protocol access rights.

Rows

One per access right.

Columns

Column name:	Data type:	Description:
ACCESS_RIGHT	VARCHAR2(20)	Name of the access right.
PROTOCOL-_TYPE	NUMBER(10)	Type of protocol: –1 — Non-protocol 1 — Protocol 2 — Resolve
ABBREV	VARCHAR2(4)	Abbreviation of the access right.
NUM_ROLES	NUMBER(3)	Number of Oracle roles associated with the access right.

Index

Index name:	Unique?:	Columns indexed:
ACCESS_RIGHT_PK	Yes	ACCESS_RIGHT

ACTIVITY_LOG

The CTS.ACTIVITY_LOG table stores information about the use of menu commands.

Rows

One per use of a menu command.

Columns

Column name:	Data type:	Description:
ACTIVITY_ID	NUMBER(10)	Unique identifier of the activity log record.
ACTIVITY_DATE	DATE	Date and time at which the activity occurred.
USERNAME	VARCHAR2(20)	User account that performed the activity.
PROTOCOL	VARCHAR2(20)	Name of the protocol in which the activity occurred.
MODULE_NAME	VARCHAR2(10)	Three-letter identifier of the Clintrial software module in which the activity occurred.
MENU_NAME	VARCHAR2(80)	Internal name of the menu.
MENU_TEXT	VARCHAR2(40)	Name of the menu command.

Index

Index name:	Unique?:	Columns indexed:
ACTIVITY_LOG_PK	Yes	ACTIVITY_ID

CATDEFS

The CTS.CATDEFS table stores information about flag categories and note categories.

Rows

One per flag category or note category.

Columns

Column name:	Data type:	Description:
CATNAME	VARCHAR2(20)	Name of the flag category or note category.
DESCRIPTION	VARCHAR2(240)	Description of the flag category or note category.
STATUS	VARCHAR2(10)	Status of the flag category or note category: • OK • DELETED
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.
LOCK_STATUS	NUMBER(1)	0 — The flag category or note category is modifiable. 1 — The flag category or note category is not modifiable, but it can be reset to modifiable. 2 — The flag category or note category is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete

Index

Index name:	Unique?:	Columns indexed:
CATDEFS_PK	Yes	CATNAME

CC_DICTIONARY

The CTS.CC_DICTIONARY table stores information about Central Coding Dictionaries.

Rows

One per dictionary.

Columns

Column name:	Data type:	Description:
DICT_ID	NUMBER(15)	Internal Identifier
DICT_NAME	VARCHAR2(20)	Name of the dictionary in Central Coding
DICT_VERSION	VARCHAR2(30)	Version of the dictionary
DICT_CULTURE	VARCHAR2(10)	Language used in the dictionary. Must be a value in the CTS_LANGUAGE codelist.
MODDATE	TIMESTAMP	Date of creation or modification
MODUSER	VARCHAR2(20)	User who made the modification
STATUS	VARCHAR2(10)	‘OK’ ’DELETED’ ‘CREATED’

Index

Index name:	Unique?:	Columns indexed:
CC_DICTIONARY_PK	Yes	DICT_ID

CC_DICTIONARY_LABEL

The CTS.CC_DICTIONARY_LABEL table stores information about Labels used by Central Coding dictionaries.

Rows

One per dictionary label.

Columns

Column name:	Data type:	Description:
DICT_ID	NUMBER(15)	Internal Identifier
CC_LABEL_TYPE	NUMBER(2)	1 — Target 2 — Associated 3 — Verbatim Type
CC_LABEL_NAME	VARCHAR2(64)	Label used in Central Coding for this item.

Index

Index name:	Unique?:	Columns indexed:
CC_DICT_LABEL_PK	Yes	DICT_ID, CC_LABEL_NAME

CTS_PROTOCOLS

The CTS.CTS_PROTOCOLS table stores information about protocols.

Rows

One per protocol.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
DESCRIPTION	VARCHAR2(240)	Description of the protocol.
CREATOR	VARCHAR2(20)	User account that created the protocol.
CREATE_DATE	DATE	Date and time at which the protocol was created.
CREATE_ITEM-_PRIV	VARCHAR2(5)	Obsolete
MOD_SRCHLST-_PRIV	VARCHAR2(5)	0 — The searchlist cannot be modified. 1 — The searchlist can be modified.
SEARCHLIST	VARCHAR2(240)	Obsolete.
SITE	VARCHAR2(20)	Name of the database from which the protocol was imported.
IMPORT_DATE	DATE	Date and time at which the protocol was imported.
STATUS	VARCHAR2(20)	Status of the protocol: • CREATING • DELETED • NORMAL • PT_CHKD • PT_DD_INSTALLED • PT_EMPTY • PT_ERROR • PT_INSTALLED • PT_LOADED • PT_OK • PT_UNCHKD
PROT_NUM	NUMBER(15)	Unique identifier assigned by the Clintrial software to the protocol.

Column name:	Data type:	Description:
DATASPACE	VARCHAR2(30)	Name of the tablespace to be used for database tables for the protocol.
INDEXSPACE	VARCHAR2(30)	Name of the tablespace to be used for indexes for database tables for the protocol.
PROT_UID	NUMBER(9)	Obsolete.
AUDIT-_SPONSOR	NUMBER(1)	0 — Sponsor notes are not audited for the protocol. 1 — Sponsor notes are audited for the protocol.
AUDIT_INVEST	NUMBER(1)	0 — Investigator notes are not audited for the protocol. 1 — Investigator notes are audited for the protocol.
VIEW-_PROTOCOL	NUMBER(1)	Obsolete.
BASE-_PROTOCOL	VARCHAR2(20)	For a view protocol, the name of the base protocol.
CPDATE	DATE	For a view protocol, the checkpoint date.
CONDITION	LONG	Obsolete.
AUDIT_COM-MENCE_DEF	VARCHAR2(20)	Default audit start point for panels in the protocol: • ENTRY • VERIFICATION • VALIDATION • VALIDITY • MERGE

Column name:	Data type:	Description:
MOD_AUDIT- _PRIV	VARCHAR2(1)	0 — The audit start point cannot be set on a panel-by-panel basis for the protocol. 1 — The audit start point can be set on a panel-by-panel basis for the protocol.
DB_ID	NUMBER(5)	Obsolete.
TYPE	NUMBER(10)	Type of protocol: 1 — Clinical Data 2 — Coding Thesaurus 3 — View Protocol 4 — Lab Loader 5 — Clintrace Storage Area
VERSION	VARCHAR2(10)	Number of the Clintrial software release.
PROC- _LANGUAGE	VARCHAR2(6)	Name of the language in which data-entry processing procedures and validation procedures are written.
PARENT- _PROTOCOL	VARCHAR2(20)	Protocol used as a basis for this protocol's objects.
MOD_OBJECTS- _PRIV	NUMBER(1)	Reserved for future use.
ENROLL_PANEL	VARCHAR2(20)	Name of the enrollment panel for the protocol.
SUBJECT_ITEM	VARCHAR2(20)	Name of the subject item for the protocol.
BLOCK_ITEM	VARCHAR2(20)	Name of the block key item for the protocol.
PAGE_ITEM	VARCHAR2(20)	Name of the page key item for the protocol.

Column name:	Data type:	Description:
BLOCK_REPEAT-_ITEM	VARCHAR2(20)	Name of the block repeat key item for the protocol.
PAGE_REPEAT-_ITEM	VARCHAR2(20)	Name of the page repeat key item for the protocol.
ERRORLOG-_ITEM	VARCHAR2(20)	Name of the additional Error Log item, which is shown in the Error Log report.
ENCODE-_OVERRIDE	NUMBER(1)	0 — Automatic coding does not override interactive coding. 1 — Automatic coding overrides interactive coding.
VIEW_OBJECT-_ID	NUMBER(15)	For a view protocol, identifier of the text object that contains the view restriction clause in CTSDD.OBJINDX.
HELP_FILE	VARCHAR2(240)	Name of the site-defined Help file attached to the protocol.
HELP_CONTEXT-_POINT	NUMBER(5)	Context point number within the user-defined help file.
LOCKED	NUMBER(1)	0 — The protocol is not locked. 1 — The protocol is locked.
DATA_DICT	NUMBER(1)	0 — The protocol is not a data dictionary protocol. 1 — The protocol is a data dictionary protocol.

Column name:	Data type:	Description:
EDC_PROTOCOL	NUMBER(2)	This column contains information related to EDC protocols • Whether the protocol can contain paper data, EDC data or both (Hybrid). • Whether data dictionary changes have been made to the protocol since the last synchronization from CIS. • Whether panel metadata changes are allowed to be made to an EDC protocol, even though data has already been collected. 1 — Only paper data can be stored within this protocol. No dictionary modifications have been made. 2 — Only EDC data can be stored within this protocol. No dictionary modifications have been made. 3 — Both paper and EDC data can be stored within this protocol (Hybrid). No dictionary modifications have been made, and panel modifications are not allowed. 5 — Only paper data can be stored within this protocol. Dictionary modifications have been made. 6 — Only EDC data can be stored within this protocol. Dictionary modifications have been made, and panel modifications are not allowed. 7 — Both paper and EDC data can be stored within this protocol (Hybrid). Dictionary modifications have been made, and panel modifications are not allowed. 10 — Only EDC data can be stored within this protocol. No dictionary modifications have been made, and panel modifications are allowed.

Column name:	Data type:	Description:
		11 — Both paper and EDC data can be stored within this protocol (Hybrid). No dictionary modifications have been made, and panel modifications are allowed. 14 — Only EDC data can be stored within this protocol. Dictionary modifications have been made, and panel modifications are allowed. 15 — Both paper and EDC data can be stored within this protocol (Hybrid). Dictionary modifications have been made, and panel modifications are allowed.

Index

Index name:	Unique?:	Columns indexed:
CTS_PROTOCOLS_PK	Yes	PROTOCOL

CTS_USERGROUPS

The CTS.CTS_USERGROUPS table stores information about the user accounts that are part of usergroups.

Rows

One per user account in a usergroup.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup.

Column name:	Data type:	Description:
USERNAME	VARCHAR2(20)	Name of the user account.

Index

Index name:	Unique?:	Columns indexed:
CTS_USER-GROUPS_PK	Yes	USERGROUP, USERNAME

CTS_USERGROUPS_AUDIT

The CTS.CTS_USERGROUPS_AUDIT table stores information about modifications to usergroups.

Rows

One row for every creation or deletion of a usergroup.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup.
MODDATE	TIMESTAMP	Date of the modification.
MODUSER	VARCHAR2(20)	User who made the modification.
ACTION	VARCHAR2(2)	I=Inserted, D=Deleted

Index name:	Unique?:	Columns indexed:
CTS_USER-GROUPS_AUDIT_PK	Yes	USERGROUP, MODDATE

CTS_USER

The CTS.CTS_USER table stores information about Clintrial software users.

Rows

One per user.

Columns

Column name:	Data type:	Description:
USERNAME	VARCHAR2(20)	Name of the user account.
DESCRIPTION	VARCHAR2(80)	Description of the user.
FULL_NAME	VARCHAR2(80)	Complete name of the user.
CREATED	DATE	Date the user account was created.

Index

Index name:	Unique?:	Columns indexed:
CTS_USER_PK	Yes	USERNAME

CTS_USER_AUDIT

The CTS.CTS_USER_AUDIT table stores information about modifications to user accounts.

Rows

One for each modification of a user account (Insert, Update, Delete).

Columns

Column name:	Data type:	Description:
USERNAME	VARCHAR2(20)	Name of the user account.
DESCRIPTION	VARCHAR2(80)	Description of the user.
FULL_NAME	VARCHAR2(80)	Complete name of the user.
CREATED	DATE	Date the user account was created.
DEFAULT_TABLESPACE	VARCHAR2(30)	Default tablespace for account.
TEMPORARY_TABLESPACE	VARCHAR2(30)	Temporary tablespace for account.
PROFILE	VARCHAR2(30)	Oracle profile for the account.
MODDATE	TIMESTAMP	Date of modification.
MODUSER	VARCHAR2(20)	user that made the modification.
ACTION	VARCHAR2(2)	I=Inserted, U=Updated, D=Deleted

Index

Index name:	Unique?:	Columns indexed:
CTS_USER_AUDIT_PK	Yes	USERNAME, MODDATE

DATABASE

The CTS.DATABASE table stores information about Clintrial software database instances.

Rows

One per database instance.

Columns

Column name:	Data type:	Description:
NAME	VARCHAR2(20)	Unique name for the instance.
DESCRIPTION	VARCHAR2(240)	Description of the instance.
DB_NAME	VARCHAR2(40)	Global database name for the instance.
DB_ID	NUMBER(5)	Unique identifier for the instance.
SQLNET	VARCHAR2(60)	Oracle Net Service Name.
TYPE	NUMBER(2)	0 — Nonlocal 1 — Local (this instance)
DISTR_SRC	NUMBER(1)	0 — The database is not a source for Multisite distribution. 1 — The data is a source for Multisite distribution.
DISTR_DEST	NUMBER(1)	0 — The database is not a destination for Multisite distribution. 1 — The database is a destination for Multisite distribution.

Index

Index name:	Unique?:	Columns indexed:
DATABASE_PK	Yes	NAME

EXCEPTION_MESSAGE

The CTS.EXCEPTION_MESSAGE table stores information about messages that display in the Clintrial software.

Rows

One per message.

Columns

Column name:	Data type:	Description:
EXCEPTION- _MESSAGE_ID	NUMBER	Unique identifier of the message.
LANGUAGE_ID	NUMBER	Language of the message: 1 — English
USER_DS	VARCHAR2(255)	Message text that is displayed to the user.
TECHNICAL_DS	VARCHAR2(255)	Message text that is sent to the CLINTRIA.LOG file.
LOG_FL	NUMBER	0 — Do not send to the log file. 1 — Send to the log file.
BUTTON- _MESSAGE_NR	NUMBER	Number indicating buttons displayed in the message dialog box.
EXCEPTION- _ICON_NR	NUMBER	Number indicating the icons displayed in the message dialog box.
EXCEPTION- _DEFAULT_NR	NUMBER	Number indicating the default button.
HELPFILE	VARCHAR2(40)	Reserved.
HELPKEYWORD	VARCHAR2(255)	Number indicating additional help (a context point) for the message.

Index

Index name:	Unique?	Columns indexed:
EXCEPTION_MESSAGE-_PK	Yes	EXCEPTION_MESSAGE_ID, LANGUAGE_ID

INVESTIGATOR_SITE

The CTS.INVESTIGATOR_SITE table stores information about investigator sites.

Rows

One per investigator site.

Columns

Column name:	Data type:	Description:
NAME	VARCHAR2(20)	Unique name of the site at which the investigator enters data.
DESCRIPTION	VARCHAR2(80)	Description of the site.
TIME_ZONE	VARCHAR2(10)	Time zone in which the investigator site is located.
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.

Index

Index name:	Unique?:	Columns indexed:
INVESTIGATOR-_SITE_PK	Yes	NAME

JOB_LOG

The CTS.JOB_LOG table stores information about batch jobs run in the Clintrial software.

Rows

One per execution or submission of a batch job.

Columns

Column name:	Data type:	Description:
JOB_ID	NUMBER(10)	Unique identifier of the job.
JOB_TYPE	VARCHAR2(10)	Type of job: • AUDIT_RPT • CODE • GLOBAL CHG • GLOBAL DEL • MERGE • SCREEN • TRANSFER • VALIDATE ***Note:*** AUDIT_RPT is the data generation step of the Subject Audit Report. ***Note:*** TRANSFER is Lab Loader Transfer
PROTOCOL	VARCHAR2(20)	Name of the protocol.

Column name:	Data type:	Description:
PANEL	VARCHAR2(20)	Name of the panel.
DBTABLE	VARCHAR2(6)	Name of the database table for which the process occurred.
START- _DATETIME	DATE	Date and time at which the process started.
END_DATETIME	DATE	Date and time at which the process ended.
LOG_USER	VARCHAR2(20)	User name that ran or submitted the job.
NUM_SELECTED	NUMBER(10)	Total number of records processed.
NUM_SUCCESS	NUMBER(10)	Total number of records successfully processed.
NUM_FAILURE	NUMBER(10)	Total number of records for which there were processing errors. If the JOB_TYPE is VALIDATE, this is the total number of failed rules with the error action REJECT. If the JOB_TYPE is SCREEN, this is the total number of records that failed screening and for which data checks were not overridden.
NUM_WARNING	NUMBER(10)	If the JOB_TYPE is VALIDATE, the total number of REPORT errors. If the JOB_TYPE is SCREEN, the number of records that failed screening.
OVERALL- _STATUS	VARCHAR2(10)	NORMAL — Completed with no errors. ERROR — Completed with errors. RUNNING — Still running.
BATCH_ID	NUMBER(10)	Identifier of the job.
REMARKS	VARCHAR2(240)	COMPLETED or error message.
RESTRICTION	VARCHAR2(2000)	Restriction clause used to select records for the job.

Indexes

Index name:	Unique?:	Columns indexed:
JOB_LOG_IDX	No	PROTOCOL, JOB_TYPE
JOB_LOG_PK	Yes	JOB_ID

OBJINDX for Oracle parameters

The CTS.OBJINDX table stores information about Oracle parameters and Clintrial software parameters. This section describes the CTS.OBJINDX table for Oracle parameters.

Rows

One per type of database table or index.

Columns

Column name:	Data type:	Description:
OBJECT	VARCHAR2(10)	Type of object: INDXPARMS — Index for database table TABPARMS — Database table
ONAME	VARCHAR2(20)	Type of table or index: • CTSCODES • CTS_AUDIT • CTS_DATA • CTS_UPDATE • ERRORLOG • SUBJECT_SPACE • TAGS_SPACE • TAGS_AUDIT_SPACE (only if ONAME is TABPARMS)

Column name:	Data type:	Description:
LASTMOD	DATE	Date and time at which the parameter was last modified.
I3	VARCHAR2(240)	Value of the Oracle parameter INITIAL.
I4	VARCHAR2(240)	Value of the Oracle parameter NEXT.
I6	VARCHAR2(240)	Value of the Oracle parameter MINEXTENTS.
I7	VARCHAR2(240)	Value of the Oracle parameter MAXEXTENTS.
I8	VARCHAR2(240)	Value of the Oracle parameter PCTINCREASE.
I9	VARCHAR2(240)	Value of the Oracle parameter PCTUSED.
I10	VARCHAR2(240)	Value of the Oracle parameter PCTFREE.
I11	VARCHAR2(240)	Value of the Oracle parameter INITRANS.
I12	VARCHAR2(240)	Value of the Oracle parameter MAXTRANS.
I13	VARCHAR2(240)	If OBJECT is TABPARMS and ONAME is CTSCODES, the name of the tablespace containing unaggregated codelists. If OBJECT is INDXPARMS and ONAME is CTSCODES, the name of the tablespace containing indexes on unaggregated codelists.

Index

Index name:	Unique?:	Columns indexed:
OBJINDX_PK	Yes	OBJECT, ONAME

OBJINDX for Clintrial software parameters

The CTS.OBJINDX table stores information about Oracle parameters and Clintrial software parameters. This section describes the CTS.OBJINDX table for Clintrial software parameters.

Rows

One per parameter.

Columns

Column name:	Data type:	Description:
OBJECT	VARCHAR2(10)	Type of Clintrial software parameter: SYS_PARAM — System parameter USER_PARAM — User preference PROT_PARAM — System parameter
ONAME	VARCHAR2(20)	Name of the parameter.
LASTMOD	DATE	Date and time at which the parameter was last modified.
I4	VARCHAR2(240)	Value of the parameter.
I5	VARCHAR2(240)	Data type of the parameter: • BOOL • DATE • FIXED • RESPONSE • SPECIAL • TEXT
I10	VARCHAR2(240)	Language of the parameter description.
I11	VARCHAR2(240)	Lower bound of the parameter.

Column name:	Data type:	Description:
I12	VARCHAR2(240)	Upper bound of the parameter.
I13	VARCHAR2(240)	Codelist associated with the parameter.
I14	VARCHAR2(240)	Version number of the parameter.
I15	VARCHAR2(240)	Identifier of the parameter group.
I25	LONG	Description of the parameter.

Index

Index name:	Unique?:	Columns indexed:
OBJINDX_PK	Yes	OBJECT, ONAME

PARAM_AUDIT

The CTS.PARAM_AUDIT table stores information about modifications to the values of system parameters, or the default values of protocol parameters.

Rows

One row for every modification of a system or default protocol parameter.

Columns

Column name:	Data type:	Description:
OBJECT	VARCHAR2(10)	SYS_PARAM or PROT_PARAM
ONAME	VARCHAR2(20)	Name of the parameter.
LASTMOD	DATE	Date of modification.

Column name:	Data type:	Description:
MODUSER	VARCHAR2(20)	User account that made the modification.
PARAM_VALUE	VARCHAR2(240)	Value of the parameter

Index

Index name:	Unique?:	Columns indexed:
PARAM_AUDIT_PK	Yes	OBJECT, ONAME, LASTMOD

PROTOCOL_LOCK_HISTORY

The CTS.PROTOCOL_LOCK_HISTORY table stores information about protocol locking and unlocking.

Rows

One per each locking or unlocking of a protocol.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
MODDATE	DATE	Date and time on which the protocol was locked or unlocked.
ACTION	NUMBER(1)	0 — The protocol was unlocked. 1 — The protocol was locked.
REASON	VARCHAR2(2000)	Reason that the protocol was locked or unlocked.

Column name:	Data type:	Description:
MODUSER	VARCHAR2(20)	User account that locked or unlocked he protocol.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PROT_LOCK_HIST_PK	Yes	PROTOCOL, MODDATE

PROTOCOL_PARAM

The CTS.PROTOCOL_PARAM table stores information about protocol parameters that have been modified to different settings than the systemwide defaults.

Rows

One per protocol parameter that has been modified to a different setting than the systemwide default.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
PARAM_NAME	VARCHAR2(50)	Name of the protocol parameter.
PARAM_VALUE	VARCHAR2(240)	Description of the protocol parameter.
DB_ID	NUMBER(5)	Obsolete.

Column name:	Data type:	Description:
MODDATE	TIMESTAMP	Date of last modification.
MODUSER	ARCHAR2(20)	User account who made the modification.

Index

Index name:	Unique?:	Columns indexed:
PROT_PARAM_PK	Yes	PROTOCOL, PARAM_NAME

PROTOCOL_PARAM_AUDIT

The CTS.PROTOCOL_PARAM_AUDIT table stores information about modifications to protocol parameters.

Rows

One for every modification to a protocol parameter.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
PARAM_NAME	VARCHAR2(50)	Name of the protocol parameter.
PARAM_VALUE	VARCHAR2(240)	Description of the protocol parameter.
MODDATE	TIMESTAMP	Date of last modification.
MODUSER	ARCHAR2(20)	User account who made the modification.

Index

Index name:	Unique?:	Columns indexed:
PROT_PARAM_AUDIT_PK	Yes	PROTOCOL, PARAM_NAME, MODDATE

REGISTRY

The CTS.REGISTRY table stores information about which Clintrial software components, as well as other Oracle product components, have been installed on the server.

Rows

One per installation of the Clintrial software core modules on the server, and one per extended module, or other component, installed on the server.

Columns

Column name:	Data type:	Description:
CT_OPTION	VARCHAR2(5)	Identifier of the component.
STATUS	NUMBER(1)	1 — Installed
MODDATE	DATE	Date and time at which the component was installed.
VERSION	VARCHAR2(10)	Version number of the component.
PATCH_LEVEL	VARCHAR2(10)	Patch number of the component, if a patch has been applied.

Index

Index name:	Unique?:	Columns indexed:
REGISTRY_PK	Yes	CT_OPTION

SEARCH_LIST

The CTS.SEARCH_LIST table stores information about protocol searchlists.

Rows

One per protocol that is in a searchlist.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol to which the searchlist is attached.
SEARCH_PROTOCOL	VARCHAR2(20)	Name of the protocol in the searchlist.
SEARCH_ORDER	NUMBER10)	Order of searching; lower numbers are searched first.
DB_ID	NUMBER(5)	Obsolete.

Indexes

Index name:	Unique?:	Columns indexed:
SEARCH_LIST_PK	Yes	PROTOCOL, SEARCH_PROTOCOL

TAGDEFS

The CTS.TAGDEFS table stores information about flag definitions and note definitions.

Rows

One per flag definition or note definition.

Columns

Column name:	Data type:	Description:
TAGID	NUMBER(15)	Unique identifier of the flag or note.
CATNAME	VARCHAR2(20)	Name of the flag category or note category.
TAGNAME	VARCHAR2(20)	Name of the flag name or note name.
DESCRIPTION	VARCHAR2(240)	Description of the flag or note.
STATUS	VARCHAR2(10)	Status of the flag definition of note definition: • OK • DELETED
DB_ID	NUMBER(5)	Unique identifier of the Clintrial software database instance.
LOCK_STATUS	NUMBER(1)	0 — The flag definition or note definition is modifiable. 1 — The flag definition or note definition is not modifiable, but it can be reset to modifiable. 2 — The flag definition or note definition is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.

Column name:	Data type:	Description:
LOCK_COPY	NUMBER(1)	Obsolete.

Indexes

Index name:	Unique?:	Columns indexed:
TAGDEFS_PK	Yes	TAGID
TAGDEFS_IDX	Yes	CATNAME, TAGNAME

USERGROUP

The CTS.USERGROUP table stores information about usergroups.

Rows

One per usergroup.

Column

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup.

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_PK	Yes	USERGROUP

USERGROUP_AUDIT

The CTS.USERGROUP table stores information about modifications to members of a usergroup.

Rows

One row for every modification to the members of a usergroup.

Column

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup.
USERNAME	VARCHAR2(20)	Name of the user in this group.
MODDATE	TIMESTAMP	Date of modification.
MODUSER	VARCHAR2(20)	user that made the modification.
ACTION	VARCHAR2(2)	I=Inserted, D=Deleted

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_AUDIT_PK	Yes	USERGROUP, USERNAME, MODDATE

USERGROUP_ACCESS

The CTS.USERGROUP_ACCESS table stores information about access rights and levels granted to users and usergroups.

Rows

One per access right granted.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the user or usergroup.
PROTOCOL	VARCHAR2(20)	Name of the protocol.
ACCESS_RIGHT	VARCHAR2(20)	Name of the access right.
IS_GROUP	NUMBER(1)	0 — The value of USERGROUP is the name of a user. 1 — The value of USERGROUP is the name of a usergroup.
ACCESS_LEVEL	NUMBER(10)	Value of the access level: 1 — Read, Basic 2 — No Delete 3 — Full, Write 4 — Publish

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_ACCESS-_PK	Yes	USERGROUP, PROTOCOL, ACCESS_RIGHT, IS_GROUP

USERGROUP_ACCESS_AUDIT

The CTS.USERGROUP_ACCESS table stores information about modifications to access rights and levels granted to users and usergroups.

Rows

One row for every modification to the privileges of a user account or usergroup.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the user or usergroup. DEFAULT — NONE
PROTOCOL	VARCHAR2(20)	Name of the protocol.
ACCESS_RIGHT	VARCHAR2(20)	Name of the access right.
IS_GROUP	NUMBER(1)	0 — The value of USERGROUP is the name of a user. 1 — The value of USERGROUP is the name of a usergroup.
ACCESS_LEVEL	NUMBER(10)	Value of the access level: 1 — Read, Basic 2 — No Delete 3 — Full, Write 4 — Publish
MODDATE	TIMESTAMP	Date of modification.
MODUSER	VARCHAR2(20)	User who made the modification.

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_ACCESS_AUDIT_PK	Yes	USERGROUP, PROTOCOL, ACCESS_RIGHT, IS_GROUP, MODDATE
ACCESS_AUDIT_PROT_IDX	No	USERGROUP, PROTOCOL, ACCESS_RIGHT, IS_GROUP

USERGROUP_ACCESS_PANEL

The CTS.USERGROUP_ACCESS_PANEL table stores information about access rights granted to users or usergroups for protected panels.

Rows

One per access right granted to a user or usergroup for a protected panel.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup or user.
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the protected panel.
PANEL	VARCHAR2(20)	Name of the protected panel.
ACCESS_RIGHT	VARCHAR2(20)	Name of the access right.
IS_GROUP	NUMBER(1)	0 — The value of USERGROUP is the name of a user. 1 — The value of USERGROUP is the name of a usergroup.

Column name:	Data type:	Description:
ACCESS_LEVEL	NUMBER(10)	Value of the access level: 1 — Read, Basic 2 — No Delete 3 — Full, Write 4 — Publish

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_ACCESS-_PANEL_PK	Yes	USERGROUP, PROTOCOL, PANEL, ACCESS_RIGHT, IS_GROUP

USERGROUP_ACCESS_PANEL_AUDIT

The CTS.USERGROUP_ACCESS_PANEL_AUDIT table stores information about modifications to access rights and levels granted to users and usergroups for protected panels.

Rows

One row for every modification of the protected panel access levels for a user or usergroup.

Columns

Column name:	Data type:	Description:
USERGROUP	VARCHAR2(20)	Name of the usergroup or user.
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the protected panel.

Column name:	Data type:	Description:
PANEL	VARCHAR2(20)	Name of the protected panel.
ACCESS_RIGHT	VARCHAR2(20)	Name of the access right.
IS_GROUP	NUMBER(1)	0 — The value of USERGROUP is the name of a user. 1 — The value of USERGROUP is the name of a usergroup.
ACCESS_LEVEL	NUMBER(10)	Value of the access level: 1 — Read, Basic 2 — No Delete 3 — Full, Write 4 — Publish
MODDATE	TIMESTAMP	Date of modification.
MODUSER	VARCHAR2(20)	User that made the modification.

Index

Index name:	Unique?:	Columns indexed:
USERGROUP_ACCESS_PANEL_AUDIT_PK	Yes	USERGROUP, PROTOCOL, PANEL, ACCESS_RIGHT, IS_GROUP, MODDATE

USER_PARAM

The CTS.USER_PARAM table stores information about user preferences that have been modified to a different setting than the system-wide default.

Rows

One per user preference that has been modified to a different setting than the systemwide default.

Columns

Column name:	Data type:	Description:
PARAM_TYPE	VARCHAR2(10)	Reserved.
PARAM_NAME	VARCHAR2(20)	Name of the user preference.
USERNAME	VARCHAR2(20)	User account that modified the user preference.
PARAM_VALUE	VARCHAR2(2000)	Value of the user preference.

Index

Index name:	Unique?:	Columns indexed:
USER_PARAM_PK	Yes	USERNAME, PARAM_NAME, PARAM_TYPE

4 CTPROC Account

Note: This account has not been documented previously, because there were not any tables.

SUBJECT_AUDIT_RECORD

The **CTPROC.SUBJECT_AUDIT_RECORD** table stores information about modifications to subjects included in a Subject Audit Report.

One row for each modification to a record included in a Subject Audit report.

Columns

Column name:	Data type:	Description:
AUDIT_ID	NUMBER(1)	Unique identifier of a record included in the audit report.
JOB_ID	NUMBER(1)	Identifier of the report.
PROTOCOL	VARCHAR2(20)	Name of the protocol.
SUBJECT_ID	NUMBER(1)	Subject identifier.
PANEL	VARCHAR2(20)	Name of the panel.
CT_RECID	VARCHAR2(40)	Record identifier (in the panel).
MODDATE	DATE	Date record was modified.
MODUSER	VARCHAR2(20)	User that modified the record.
CTS$REASON	VARCHAR2(2000)	Reason for change.
SUBJECT_KEY	VARCHAR2(40)	Value of subject key.
BLOCK_KEY	VARCHAR2(40)	Value of block key.
BLOCK_REPEAT_KEY	VARCHAR2(40)	Value of block repeat key.

Column name:	Data type:	Description:
PAGE_KEY	VARCHAR2(40)	Value of page key.
PAGE_REPEAT_KEY	VARCHAR2(40)	Value of page repeat key.

Index

Index name:	Unique?:	Columns indexed:
SUBJ_AUDIT_RECORD_PK	Yes	AUDIT_ID
SUBJ_AUDIT_RECORD_IDX	No	JOB_ID, PROTOCOL, SUBJECT_ID, BLOCK_KEY, BLOCK_REPEAT_KEY, PAGE_KEY, PAGE_REPEAT_KEY

SUBJECT_AUDIT_ITEM

The **CTPROC.SUBJECT_AUDIT_ITEM** table stores information about data value changes for subjects included in a Subject Audit Report.

One row for each data item modified in records included in a Subject Audit Report.

Columns

Column name:	Data type:	Description:
AUDIT_ID	NUMBER(1)	Unique identifier of a record included in the audit report.
ITEM_NAME	VARCHAR2(20)	Name of an item.
PREV_VALUE	VARCHAR2(2000)	Value of the item before modification.
NEW_VALUE	VARCHAR2(2000)	Value of the item after modification.

Index

Index name:	Unique?:	Columns indexed:
SUBJ_AUDIT_ITEM_PK	Yes	AUDIT_ID, ITEM_NAME

5 CTSCODES Account

AGGREGATED_CODES

The CTSCODES.AGGREGATED_CODES table stores the contents of all aggregated codelists in the database instance.

Rows

One per combination of codelist and value.

Columns

Column name:	Data type:	Description:
CODELIST	VARCHAR2(20)	Name of the codelist.
VARNO	NUMBER(2)	This value is always 1.
CODE	VARCHAR2(80)	Code
VALUE	VARCHAR2(80)	Value associated with the code.
LABEL	VARCHAR2(80)	Short descriptive label associated with the code.
LONGLABEL	VARCHAR2(240)	Long descriptive label associated with the code.
DB_ID	NUMBER(5)	Obsolete
STATUS	NUMBER(1)	Status of the codelist entry: 0 — Valid -1 — Invalid
CODE_ORDER	NUMBER(5)	Position of the codelist entry in the ordered set of codelist entries.

Column name:	Data type:	Description:
SUBSET_REQD	NUMBER(1)	0 — The codelist entry is not required in subset codelists that are based on this codelist. 1 — The codelist entry is included (regardless of the subset restriction) in the subset codelists that are based on this codelist.
SUBSET_VALUE	NUMBER(5)	Subset value, which may be used to create subset codelists.

Index

Index name:	Unique?:	Columns indexed:
AGGREGATED_CODES-_PK	Yes	CODELIST, VARNO, CODE, VALUE

CODE_INDEX

The CTSCODES.CODE_INDEX table stores the attributes of codelists.

Rows

One per codelist.

Columns

Column name:	Data type:	Description:
CODELIST	VARCHAR2(20)	Name of the codelist.
CHECKTYPE	VARCHAR2(10)	Obsolete.

Column name:	Data type:	Description:
AGGREGATED	VARCHAR2(10)	NO — The codelist is not aggregated. YES — The codelist is aggregated. VIEW — The codelist is a view codelist. SUBSET — The codelist is a subset codelist.
CODETYPE	VARCHAR2(10)	Data type of the code: • FIXED • TEXT
VALUETYPE	VARCHAR2(10)	Data type of the value: • FIXED • TEXT
DESCRIPTION	VARCHAR2(80)	Description of the codelist.
MODDATE	DATE	Date and time at which the codelist was created or last modified.
AUTHOR	VARCHAR2(20)	User account that created or last modified the codelist.
SASNAME	VARCHAR2(6)	SAS format name associated with the codelist.
DB_ID	NUMBER(5)	Obsolete.
STATUS	NUMBER(1)	Status of the codelist: 0 — Valid -1 — Invalid
DATA_DICT	NUMBER(1)	0 — The codelist is not a data dictionary codelist. 1 — The codelist is a data dictionary codelist.
BASE_CODELIST	VARCHAR2(20)	Name of the base codelist for a subset codelist.
SUBSET- _RESTRICTION	VARCHAR2(2000)	Subset restriction clause (Oracle WHERE clause for a subset codelist.

Column name:	Data type:	Description:
VIEW_CREATED	NUMBER(1)	0 — The codelist's view has not been created or is invalid. 1 — The codelist's view has been created. Null — The codelist is not a view codelist or subset codelist.

Indexes

Index name:	Unique?:	Columns indexed:
CODESAS_INDEX	Yes	SASNAME
CODE_INDEX_PK	Yes	CODELIST

CODELIST_ASSOC

The CTSCODES.CODELIST_ASSOC table stores information about the association of imported codelists to existing codelists at the receiving site.

Rows

One per association of an imported codelist to an existing codelist.

Columns

Column name:	Data type:	Description:
IMPORT_NUM	NUMBER(6)	Unique identifier of the codelist import.
SOURCE_NAME	VARCHAR2(20)	Name of the codelist at the sending site.
INSTALLED-_NAME	VARCHAR2(20)	Name of the codelist at the receiving site.

Index

Index name:	Unique?:	Columns indexed:
CODELIST_ASSOC_PK	Yes	IMPORT_NUM, SOURCE_NAME

IMPORT_LOG

The CTSCODES.IMPORT_LOG table stores information about imports of codelists.

Rows

One per import of a codelist.

Columns

Column name:	Data type:	Description:
IMPORT_NUM	NUMBER(6)	Unique identifier of the codelist import.
SOURCE_SITE	VARCHAR2(30)	Name of the database from which the codelist was exported.
EXPORT-_DATETIME	DATE	Date and time at which the codelist was exported.
IMPORT-_DATETIME	DATE	Date and time at which the codelist was imported.
CHANGECOUNT	NUMBER(8)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
IMPORT_LOG_PK	Yes	IMPORT_NUM

VIEW_CODELIST

The CTSCODES.VIEW_CODELIST table stores information about view codelists.

Rows

One per view codelist.

Columns

Column name:	Data type:	Description:
CODELIST	VARCHAR2(20)	Name of the view codelist.
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel for the view onto which the view codelist provides a view.
PANEL	VARCHAR2(20)	Name of the panel onto which the view codelist provides a view.
CODE_ITEM	VARCHAR2(20)	Name of the column that stores codes.
VALUE_ITEM	VARCHAR2(20)	Name of the column that stores values.
LABEL_ITEM	VARCHAR2(20)	Name of the column that stores short labels.
LONGLABEL-_ITEM	VARCHAR2(20)	Name of the column that stores long labels.

Column name:	Data type:	Description:
RESTRICTION	VARCHAR2(240)	SQL restriction clause that selects data from the base table to create the view codelist.
STATUS	NUMBER(2)	0 — The view is invalid or was not created. 1 — The view was created and is valid.
DB_ID	NUMBER(5)	Obsolete.
STATUS_ITEM	VARCHAR(20)	Name of the column that stores the status of the codelist entry; analogous to the STATUS column of the AGGREGATED_CODES table.
CODE_ORDER-_ITEM	VARCHAR2(20)	Name of the column that stores the position of the codelist entry; analogous to the CODE_ORDER column of the AGGREGATED_CODES table.
SUBSET_REQD-_ITEM	VARCHAR2(20)	Name of the column that stores an indication of whether the codelist entry is required; analogous to the SUBSET_REQD column of the AGGREGATED_CODES table.
SUBSET_VALUE-_ITEM	VARCHAR2(20)	Name of the column that stores the subset value; analogous to the SUBSET_VALUE column in the AGGREGATED_CODE table.
BASE_TYPE	NUMBER(1)	0 — The codelist is based on a Clintrial software panel's database tables. 1 — The codelist is based on a non-Clintrial Oracle table.

Index

Index name:	Unique?:	Columns indexed:
VIEW_CODELIST_PK	Yes	CODELIST

6 CTCLASSIFY Account

GCT_CC_ID

The CTCLASSIFY.GCT_CC_ID table supplies the drop-down list of the GCT_CC_DB_ID column in the editor for the GCT_CTX_LOC table.

A row is created in this table when the **Autocoding Sites** command on Classify's **Configuration** menu is used.

Rows

One per coding center.

Columns

Column name:	Data type:	Description:
GCT_CREATE_DT	DATE	Date on which the record was created.
LASTMOD	DATE	Date on which the record was last modified.
MODUSER	VARCHAR2(30)	User account that created or last modified the record.
GCT_SITE_NAME	VARCHAR2(40)	Coding center site name.
GCT_CC_DB_ID	NUMBER	Database identifier of the coding center.
DB_ID	NUMBER	Database identifier of the owner of the coding center.

Indexes

Index name:	Unique?:	Columns indexed:
GCT_CC_ID_PK	Yes	GCT_SITE_NAME

Index name:	Unique?:	Columns indexed:
GCT_CC_ID_UK	Yes	GCT_CC_DB_ID

GCT_CC_OMISSION

The CTCLASSIFY.GCT_CC_OMISSION table contains additional information for omission records.

A row is created in this table when a clinical data record fails automatic coding. Classify users delete omission records, and their corresponding rows in this table, by purging.

Rows

One per omission record.

Columns

Column name:	Data type:	Description:
GCT_OMISSION-_ID	VARCHAR2(20)	Unique identifier of the related GCT_DC_OMISSION row. Joins a GCT_CC_OMISSION row to a GCT_DC_OMISSION row.
GCT_SOLUTION-_ID	VARCHAR2(20)	0 (zero) if not initialized from a reusable solution, otherwise, the solution ID of the reusable solution used for the initialization.
LASTMOD	DATE	The date this row was last modified.
MODUSER	VARCHAR2(30)	User account that last modified this row.

Column name:	Data type:	Description:
GCT_SOLUTION-_TYPE	VARCHAR2(20)	Indicates the type of solution proposed or applied to the omission record: • Synonym • Verbatim • Request • None
GCT_DETAILS	VARCHAR2(256)	Used for omission records with a solution type of Request or Verbatim only. • For Request solutions, contains the text of the discrepancy message or item flag. • For Verbatim solutions, contains the new verbatim text.
GCT_SYNONYM-_ERROR	VARCHAR2(256)	Used for omission records with a solution type of Synonym only. If a synonym is proposed, but an error is received when you attempt to accept it, this item stores the error message.
GCT_CC_STATUS	VARCHAR2(20)	Indicates the internal status of the solution for the omission record: • Needs Proposal • Proposed • Accepted
GCT_DC_DB_ID	NUMBER	Database identifier of the data center.
DB_ID	NUMBER	Database identifier of the coding center.
GCT_REASON	VARCHAR2(20)	

Indexes

Index name:	Unique?:	Columns indexed:
GCT_CC_PK	Yes	GCT_OMISSION_ID
GCT_CC_IDX_A1	No	GCT_SOLUTION_ID

GCT_CTX_LOC

The CTCLASSIFY.GCT_CTX_LOC table identifies which coding center owns an omission, as determined by the data center where it was created, the protocol containing the clinical data, and the thesaurus used for coding.

To enter or modify this table, use Classify's **Mapping** command on the **Configuration** menu.

Rows

One per omission.

Columns

Column name:	Data type:	Description:
GCT_CREATE-_DATE	DATE	Date on which the record was created.
LAST_MOD	DATE	Date on which the record was last modified.
MODUSER	VARCHAR2(30)	User account that created or last modified the record.
GCT_CC_DB_ID	NUMBER)	Database identifier of the coding center.
GCT_DC_DB_ID	NUMBER	Database identifier of the data center, or a wildcard for all database identifiers.
GCT_THESAURUS	VARCHAR2(20)	Thesaurus protocol, or a wildcard for all thesaurus protocols.
GCT_PROTOCOL	VARCHAR2(20)	Clinical data protocol, or a wildcard for all clinical data protocols.

Column name:	Data type:	Description:
DB_ID	NUMBER	Database identifier of the owner of the omission.

Index

Index name:	Unique?:	Columns indexed:
GCT_CTX_LOC_PK	Yes	GCT_DC_DB_ID, GCT_THESAURUS, GCT_PROTOCOL

GCT_DC_ID

The CTCLASSIFY.GCT_DC_ID table supplies the drop-down list for the GCT_DC_DB_ID column in the editor for the GCT_CTX_LOC table.

A row is created in this table when the **Omission Handling Sites** command on Classify's **Configuration** menu is used.

Rows

One per data center.

Columns

Column name:	Data type:	Description:
GCT_CREATE_DT	DATE	Date on which the record was created.
LASTMOD	DATE	Date on which the record was last modified.
MODUSER	VARCHAR2(30)	User account that created or last modified the record.

Column name:	Data type:	Description:
GCT_SITE_NAME	VARCHAR2(40)	Data center site name.
GCT_DC_DB_ID	NUMBER	Data center database identifier.
DB_ID	NUMBER	Database identifier of the owner of the data center.
GCT_REFRESH-_DID_DT	DATE	Date on which an attempt was last made to refresh the GCT_DC_PROTOCOL table.
GCT_REFRESH-_CHANGE_DT	DATE	Reserved.

Indexes

Index name:	Unique?:	Columns indexed:
GCT_DC_ID_PK	Yes	GCT_SITE_NAME
GCT_DC_ID_UK	Yes	GCT_DC_DB_ID

GCT_DC_OMISSION

The CTCLASSIFY.GCT_DC_OMISSION table contains information about the omission records created when a clinical data record fails automatic coding in Manage.

A row is created in this table when a clinical data record fails automatic coding. Classify users delete omission records, and their corresponding rows in this table, by purging.

Rows

One per omission record.

Columns

Column name:	Data type:	Description:
GCT_OMISSION-_ID	VARCHAR2(20)	Unique identifying number for an omission record. Created by concatenating the next sequential value and DB_ID in the format *value.DB_ID*.
GCT_CREATE_DT	DATE	The date this row was first inserted into the GCT_DC_OMISSION table.
LASTMOD	DATE	The date this row was last modified.
MODUSER	VARCHAR2(30)	User account that last modified this row.
GCT_THESAURUS	VARCHAR2(20)	Coding thesaurus protocol used to code the verbatim text.
GCT_ALGORITHM	VARCHAR2(20)	Algorithm used for coding.
GCT_LANGUAGE	VARCHAR2(20)	Thesaurus language used for coding.
GCT_PROTOCOL	VARCHAR2(20)	Protocol of the clinical data record with the value that failed automatic coding.
GCT_PANEL	VARCHAR2(20)	Panel with the value that failed automatic coding.
GCT_ORCTABLE	VARCHAR2(10)	Oracle database table where the clinical data record was located at the time automatic coding failed (that is, update or data).
GCT_CODE1_ITEM	VARCHAR2(20)	Name of the Code1 item.
GCT_VERBATIM-_ITEM	VARCHAR2(20)	Name of the Verbatim Text item. (The field in which the term that could not be coded was entered.)
GCT_ERR_RECID	VARCHAR2(40)	Ct_Recid of the clinical data record that failed automatic coding.

Column name:	Data type:	Description:
GCT_SUBJECT	VARCHAR2(200)	Value of the subject context item from the clinical data record that failed automatic coding. Null for Type 0 panels.
GCT_BLOCK	VARCHAR2(20)	Value of the block context item from the clinical data record that failed automatic coding. Null for Type 0 panels.
GCT_PAGE	VARCHAR2(20)	Value of the page context item from the clinical data record that failed automatic coding. Null for Type 0 panels.
GCT_VERBATIM	VARCHAR2(256)	The text that could not be coded.
GCT_NORMAL- _VERBATIM	VARCHAR2(256)	The normalized text resulting from applying a thesaurus algorithm to the verbatim text.
GCT_DC_STATUS	VARCHAR2(20)	Indicates the status of the omission record at the implementation level: • Active • Applied • Autoresolved
GCT_CC_DB_ID	NUMBER	Database identifier of the coding center.
DB_ID	NUMBER	Database identifier of the data center.
GCT_ERROR	VARCHAR2(256)	Error text.
GCT_REASON	VARCHAR2(20)	

Indexes

Index name:	Unique?:	Columns indexed:
GCT_DC_PK	Yes	GCT_OMISSION_ID
GCT_DC_IDX_A1	No	GCT_ERR_RECID GCT_CODE1_ITEM

Index name:	Unique?:	Columns indexed:
GCT_DC_IDX_A2	No	GCT_VERBATIM, GCT_THESAURUS, GCT_ALGORITHM, GCT_LANGUAGE

GCT_DC_PROTOCOL

The CTCLASSIFY.GCT_DC_PROTOCOL table supplies the drop-down list for the thesaurus and protocol in the GCT_CTX_LOC table.

Rows

One per protocol.

Columns

Column name:	Data type:	Description:
GCT_CREATE_DT	DATE	Date on which the record was created.
LASTMOD	DATE	Date on which the record was last modified.
MODUSER	VARCHAR2(30)	User account that created or last modified the record.
GCT_DC_DB_ID	NUMBER	Data center database identifier.
GCT_THESAURUS	VARCHAR2(20)	Thesaurus protocol, or a wildcard for all thesaurus protocols.
GCT_PROTOCOL	VARCHAR2(20)	Clinical data protocol, or a wildcard for all clinical data protocols.

Column name:	Data type:	Description:
DB_ID	NUMBER	Database identifier of the owner of the protocol.

Index

Index name:	Unique?:	Columns indexed:
GCT_DC_PROTOCOL_PK	Yes	GCT_DC_DB_ID, GCT_THESAURUS, GCT_PROTOCOL

GCT_LEX_ELT

The CTCLASSIFY.GCT_LEX_ELT table defines the support elements required by Classify transformations.

Rows

One per transformation.

Columns

Column name:	Data type:	Description:
GCT_THESAURUS	VARCHAR2(20)	Coding thesaurus protocol used to code the verbatim text.
GCT_ALGORITHM	VARCHAR2(20)	Algorithm used for coding.
GCT_LANGUAGE	VARCHAR2(20)	Thesaurus language used for coding.

Column name:	Data type:	Description:
GCT_ELEMENT-_TYPE	VARCHAR2(30)	The following transformations are supported: • REPLACE_WORDS • REMOVE_CHARS • REMOVE_WORDS • PROTECTED_STEMMING
GCT_ELEMENT	VARCHAR2(200)	Text of the original phrase, word, stem, or character.
GCT_ELEMENT-_PRIME	VARCHAR2(200)	Text to replace the original phrase, word, stem, or character.
GCT_CREATE_DT	DATE	Date this row was first inserted into the GCT_LEX_ELT table.
LASTMOD	DATE	Date this row was last modified.
MODUSER	VARCHAR2(30)	User account that last modified this row.
DB_ID	NUMBER	Database identifier of the owner of the protocol.

Index

Index name:	Unique?:	Columns indexed:
GCT_LEX_ELT_PK	Yes	GCT_ELEMENT, GCT_ELEMENT_TYPE, GCT_LANGUAGE, GCT_ALGORITHM, GCT_THESAURUS

GCT_SOLUTION

The CTCLASSIFY.GCT_SOLUTION table contains information about synonym solution types when they are proposed or accepted for an omission record.

Rows

One per solution with a solution type of synonym.

Columns

Column name:	Data type:	Description:
GCT_SOLUTION-_ID	VARCHAR2(20)	Unique identifier of the related GCT_CC_OMISSION row. Joins a GCT_SOLUTION row to a GCT_CC_OMISSION row.
GCT_CREATE_DT	DATE	Date this row was first inserted into the GCT_SOLUTION table.
LASTMOD	DATE	Date this row was last modified.
MODUSER	VARCHAR2(30)	User account that last modified this row.
GCT_THESAURUS	VARCHAR2(20)	Coding thesaurus protocol to which the new synonym will be added.
GCT_ALGORITHM	VARCHAR2(20)	Algorithm to which the synonym solution applies.
GCT_LANGUAGE	VARCHAR2(20)	Thesaurus language to which the synonym solution applies.
GCT_SYNONYM	VARCHAR2(256)	Text for the new synonym.
GCT_SYNONYM-_CODE	VARCHAR2(60)	Code for the synonym; selected from an existing terms or synonyms thesaurus view.
GCT_SYNONYM-_VIEW	VARCHAR2(20)	View in which the new synonym will be created.
GCT_SYNONYM-_COMMENT	VARCHAR2(256)	Text describing the reasons for creating this synonym.

Column name:	Data type:	Description:
GCT_SOURCE-_SYNONYM	VARCHAR2(256)	Term or synonym that currently uses the selected code.
GCT_SOURCE-_VIEW	VARCHAR2(20)	Name of the thesaurus view that contains the existing term or synonym.
GCT_SOURCE-_THESAURUS	VARCHAR2(20)	Name of the coding thesaurus protocol that contains the existing term or synonym.
DB_ID	NUMBER	Database identifier of the owner of the protocol.
GCT_CC_DB_ID	NUMBER	Database identifier of the coding center.

Index

Index name:	Unique?:	Columns indexed:
GCT_SLT_PK	Yes	GCT_SOLUTION_ID

7 CTSDD Account

AUDIT_START_HISTORY

The CTSDD.AUDIT_START_HISTORY table stores information about modified audit start points.

Rows

One per modification of an audit start point.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel to which the audit start point applies.
PANEL	VARCHAR2(20)	Name of the panel to which the audit start point applies.
MODDATE	DATE	Date and time at which the audit start point was modified.
AUDIT_POINT	VARCHAR2(20)	Setting of the audit start point: • ENTRY • VERIFICATION • VALIDATION • VALIDITY • MERGE
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
AUDIT_START_HIST_PK	Yes	PROTOCOL, PANEL, MODDATE

BLOCK_REF

The CTSDD.BLOCK_REF table stores information about blocks in a study book.

Rows

One per block in a study book.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the study book that contains the block.
STUDYBOOK-_NAME	VARCHAR2(20)	Name of the study book containing the block.
BLOCK_KEY	VARCHAR2(40)	Block key value. ***Note***: For a study book for a Type 0 or 5 panel, this column stores the name of a panel.
BLOCK_TITLE	VARCHAR2(20)	Title of the block.
BLOCK_ORDER	NUMBER(5)	Order of the block in the study book.
MODDATE	DATE	Date and time at which the block was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the block.
COMPONENT_ID	NUMBER(10)	Obsolete.
DB_ID	NUMBER(5)	Obsolete.
HAS_REPEATS	NUMBER(1)	0 — The block does not allow repeats. 1 — The block allows repeats.

Column name:	Data type:	Description:
REPEATS_LIMIT	NUMBER(5)	Maximum number of block repeats allowed; if -1, there is no limit.
LOCK_STATUS	NUMBER(1)	0 — The block is modifiable. 1 — The block is not modifiable, but it can be reset to modifiable. 2 — The block is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete

Index

Index name:	Unique?:	Columns indexed:
BLOCK_REF_PK	Yes	PROTOCOL, STUDYBOOK_NAME, BLOCK_KEY

BLOCK_REF_VALUE

The CTSDD.BLOCK_REF_VALUE table stores information about other visit-related context items for which default values have been defined.

Rows

One per visit-related context item for which a default value has been defined.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.

Column name:	Data type:	Description:
STUDYBOOK-_NAME	VARCHAR2(20)	Name of the study book
BLOCK_KEY	VARCHAR2(40)	Block key value.
ITEM_NAME	VARCHAR2(20)	Name of the visit-related context item associated with the block identified by the block key value.
ITEM_VALUE	VARCHAR2(240)	Value to be used for the item specified by ITEM_NAME.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
BLOCK_REF_VALUE_PK	Yes	PROTOCOL, STUDYBOOK_NAME, BLOCK_KEY, ITEM_NAME

BLOCK_REPEATS

The CTSDD.BLOCK_REPEATS table stores information about repeating blocks used in study books.

Rows

One per repeating block.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
BLOCK_KEY	VARCHAR2(40)	Block key value.
BLOCK_REPEAT-_KEY	VARCHAR2(40)	Block repeat key value.
BLOCK_REPEAT-_ORDER	NUMBER(5)	Order of the repeating block within the series of repeating blocks. For example, if the block repeats three times, this value could be 1, 2, or 3.
IS_STATIC	NUMBER(1)	0 — The block repeat key was created during data entry. 1 — The block repeat key was predefined by the study designer.
MODDATE	DATE	Date and time at which the repeating block was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the repeating block.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
BLOCK_RPT_PK	Yes	PROTOCOL, BLOCK_KEY, BLOCK-_REPEAT_KEY

CC_TARGET

The CTSDD.CC_TARGET table stores information about coding Targets.

Rows

One per encoding target.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20) (FK)	Name of the protocol containing the panel to which the target is attached.
PANEL	VARCHAR2(20) (FK)	Name of the panel to which the target is attached.
CODE_ITEM	VARCHAR2(20)	Name of the target item that stores codes.
WORKFLOW_ITEM	VARCHAR2(20)	Name of the item that stores the workflow information. Values will be either 'SENT' or 'CODED'
DATE_ITEM	VARCHAR2(20)	Name of the item used to store the verbatim text to be coded.
VERBATIM_ITEM	VARCHAR2(20)	Name of the item used to store the Date Coded
VERBATIM_TYPE	VARCHAR2(20)	One of the verbatim types defined by the dictionary used by this target.
MODDATE	VARCHAR2(20)	Date of modification.
MODUSER	VARCHAR2(20)	User who made the modification.

Column name:	Data type:	Description:
STATUS	NUMBER(1)	Status of the coding target: Null — Newly created during panel revision. 0 — Not newly created during panel revision.
LOCK_STATUS	NUMBER(1)	0 — The coding target is modifiable. 1 — The coding target is not modifiable, but it can be reset to modifiable. 2 — The coding target is a non-modifiable copy that cannot be made modifiable except by breaking the connection.

Index

Index name:	Unique?:	Columns indexed:
CC_TARGET_PK	Yes	PROTOCOL, PANEL, CODE_ITEM

CC_TARGET_ITEM

The CTSDD.CC_TARGET_ITEM table stores information about Central Coding Target Items and Labels.

Rows

One per encoding Target Item.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20) (FK)	Name of the protocol containing the panel to which the target item is attached.
PANEL	VARCHAR2(20) (FK)	Name of the panel to which the target item is attached.
CODE_ITEM	VARCHAR2(20)	Name of the column that stores codes.
LABEL_NAME	VARCHAR2(20) L	Dictionary Label for the item
LABEL_TYPE	NUMBER(1)	Label Type for this item 1:Target; 2:Associated; 3:Verbatim Type
PANEL_NAME	VARCHAR2(20)	Name of the panel containing the item.
ITEM_NAME	VARCHAR2(20)	Name of the item, which will store/pass information with this label.

Note: In this release PANEL_NAME will always be the same as PANEL.

Index

Index name:	Unique?:	Columns indexed:
CC_TARGET_ITEM_PK	Yes	PROTOCOL, PANEL, CODE_ITEM, LABEL_NAME

DERIVATION

The CTSDD.DERIVATION table stores information about derivations.

Rows

One per derivation.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel to which the derivation is attached.
PANEL	VARCHAR2(20)	Name of the panel to which the derivation is attached.
DERIV_NAME	VARCHAR2(20)	Name of the derivation.
IS_COMPILED	NUMBER(1)	0 — The derivation has not been compiled successfully. 1 — The derivation has been compiled successfully.
MODDATE	DATE	Date and time at which the derivation was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the derivation.

Column name:	Data type:	Description:
OBJECT_ID	NUMBER(15)	Identifier of the text object that contains the text of the derivation in CTSDD-.OBJINDX.
DB_ID	NUMBER(5)	Obsolete
DESCRIPTION	VARCHAR2(240)	Description of the derivation.
LOCK_STATUS	NUMBER(1)	0 — The derivation is modifiable. 1 — The derivation is not modifiable, but it can be reset to modifiable. 2 — The derivation is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	0 — The derivation must be copied when the panel is copied. 1 — The derivation can optionally be copied when the panel is copied.

Index

Index name:	Unique?:	Columns indexed:
DERIVATION_CNST	Yes	PROTOCOL, PANEL, DERIV_NAME

DERIVATION_AUDIT

The CTSDD.DERIVATION_AUDIT table stores information about modified or deleted derivations while the panel is marked for revision.

Rows

One per modified or deleted derivation while the panel is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Same value as that of the derivation before it was modified or deleted.
PANEL	VARCHAR2(20)	Same value as that of the derivation before it was modified or deleted.
DERIV_NAME	VARCHAR2(20)	Same value as that of the derivation before it was modified or deleted.
IS_COMPILED	NUMBER(1)	Same value as that of the derivation before it was modified or deleted.
MODDATE	DATE	Date and time at which the derivation was modified or deleted.
MODUSER	VARCHAR2(20)	Date and time at which the derivation was modified or deleted.
OBJECT_ID	NUMBER(15)	Same value as that of the derivation before it was modified or deleted.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	Same value as that of the derivation before it was modified or deleted.
LOCK_STATUS	NUMBER(1)	Same value as that of the derivation before it was modified or deleted.
LOCK_COPY	NUMBER(1)	Same value as that of the derivation before it was modified or deleted.

Index

Index name:	Unique?:	Columns indexed:
DERIVATION_AUDIT-_PK	Yes	PROTOCOL, PANEL, DERIV_NAME

ENCODING_TARGET

The CTSDD.ENCODING_TARGET table stores information about coding targets.

Rows

One per coding target.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the clinical data panel that contains the item to be coded.
PANEL	VARCHAR2(20)	Name of the clinical data panel containing the item to be coded.
CODE1_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the code (or the first part of a multipart code).
CODE2_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the second part of a multipart code.
CODE3_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the third part of a multipart code.
ENCODED_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) to be coded.
WORKFLOW-_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the indication of the type of coding.
USER_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the name of the user account that performs the coding.

Column name:	Data type:	Description:
DATE_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the date and time that coding is performed.
AUTO- _MATCHES_ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the number of matches found by automatic coding.
AUTO_STEP- _ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the step of the algorithm that produces a match during automatic coding.
LANGUAGE- _ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the language used for coding.
ALGORITHM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the name of the algorithm used for automatic coding.
STATUS	NUMBER(2)	Status of the coding target: Null — Newly created during panel revision. 0 — Not newly created during panel revision.
MODDATE	DATE	Date and time at which the coding target was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the coding target.
DB_ID	NUMBER(5)	Obsolete.
NORMAL_TEXT- _ITEM	VARCHAR2(20)	Name of the item (in the clinical data panel) that stores the normalized text.
LOCK_STATUS	NUMBER(1)	0 — The coding target is modifiable. 1 — The coding target is not modifiable, but it can be reset to modifiable. 2 — The coding target is a non-modifiable copy that cannot be made modifiable except by breaking the connection.

Column name:	Data type:	Description:
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
ENCODING_TARGET-_PK	Yes	PROTOCOL, PANEL, CODE1_ITEM

ENCODING_TARGET_AUDIT

The CTSDD.ENCODING_TARGET_AUDIT table stores information about modified or deleted coding targets while the panel is marked for revision.

Rows

One per modified or deleted coding target while the panel is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
PANEL	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
CODE1_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
CODE2_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.

Column name:	Data type:	Description:
CODE3_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
ENCODED_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
WORKFLOW- _ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
USER_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
DATE_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
AUTO- _MATCHES_ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
AUTO_STEP- _ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
LANGUAGE- _ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
ALGORITHM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
STATUS	NUMBER(2)	Same value as that of the coding target before it was modified or deleted.
MODDATE	DATE	Same value as that of the coding target before it was modified or deleted.
MODUSER	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
DB_ID	NUMBER(5)	Obsolete.
NORMAL_TEXT- _ITEM	VARCHAR2(20)	Same value as that of the coding target before it was modified or deleted.
LOCK_STATUS	NUMBER(1)	Same value as that of the coding target before it was modified or deleted.

Column name:	Data type:	Description:
LOCK_COPY	NUMBER(1)	Same value as that of the coding target before it was modified or deleted.

Index

Index name:	Unique?:	Columns indexed:
ENCODING_TAUDIT-_PK	Yes	PROTOCOL, PANEL, CODE1_ITEM

ITEM

The CTSDD.ITEM table stores information about attributes of items in installed panels.

Rows

One per item for an installed panel.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel that contains the item.
PANEL	VARCHAR2(20)	Name of the panel containing the item.
ITEM_NAME	VARCHAR2(20)	Name of the item.
SASNAME	VARCHAR2(8)	SAS name of the item.
DESCRIP	VARCHAR2(240)	Description of the item.
UNITS	VARCHAR2(40)	Unit of measurement of the item.

Column name:	Data type:	Description:
DTYPE	VARCHAR2(10)	Data type of the item: • TEXT • FIXED • FLOAT • DATE • DATETIME
CODENAME	VARCHAR2(20)	Name of the codelist attached to the item.
LOOKUPNAME	VARCHAR2(20)	Name of the checklist attached to the item.
DERIVED	VARCHAR2(5)	0 — The item is not derived. 1 — The item is derived.
MANDATORY	VARCHAR2(5)	0 — The item is not required. 1 — The item is required.
RANGELB	VARCHAR2(40)	Minimum value for the item.
RANGEUB	VARCHAR2(40)	Maximum value for the item.
DBFMT	VARCHAR2(20)	Database format of the item.
IORDER	VARCHAR2(10)	Number of the column containing the item in the database tables for the panel. This value indicates the logical order of items.
THESAURUS	VARCHAR2(20)	Name of the coding thesaurus protocol used for coding the item.
CONTEXT_TYPE	NUMBER(2)	0 — Not a context item. 1 — Subject-related context item. 2 — Block-related context item. 3 — Page-related context item. 4 — Other context item.
KEY_ORDER	NUMBER(10)	If the item is part of a user-defined panel key, the order within the key.

Column name:	Data type:	Description:
MODDATE	DATE	Date and time at which the item was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the item.
LOCK_STATUS	NUMBER(1)	0 — The item is modifiable. 1 — The item is not modifiable, but it can be reset to modifiable. 2 — The item is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	0 — The item must be copied when the panel is copied. 1 — The item can optionally be copied when the panel is copied.
STATUS	NUMBER(1)	Status of the item: 0 — Valid -1 — Invalid
SORT_ORDER	NUMBER(10)	Item used for sorting within observations, and for order within the sort.
SORT_DESC	NUMBER(1)	0 — Sort is descending. 1 — Sort is ascending. This value is null if the item is not used for sorting.
SORT_IS_GRP	NUMBER(1)	0 — Item is used only for sorting, not for grouping. 1 — Item is used for sorting and for grouping of observations. This value is null if the item is not used for sorting or is not in a Type 0 panel.

Index

Index name:	Unique?:	Columns indexed:
ITEM_PK	Yes	PROTOCOL, PANEL, ITEM_NAME

ITEM_NONDD

The CTSDD.ITEM_NONDD table stores information about items in panels that are not yet installed, are deinstalled, or are marked for revision.

Rows

One per item in a panel that is not yet installed, is deinstalled, or is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Protocol containing the panel that contains the item.
PANEL	VARCHAR2(20)	Panel containing the item.
UNIQUE_NAME	VARCHAR2(20)	Name of the item.
ITEMTYPE	VARCHAR2(10)	DE — Deinstalled. UN — Not installed. PRE — Marked for revision, but the item has not been modified. REV — Marked for revision and the item has been modified.
IDDNUM	NUMBER(15)	Obsolete.
SASNAME	VARCHAR2(8)	SAS name of the item.

Column name:	Data type:	Description:
DESCRIP	VARCHAR2(240)	Description of the item.
UNITS	VARCHAR2(40)	Unit of measurement associated with the item.
DTYPE	VARCHAR2(10)	Data type of the item: • TEXT • FIXED • FLOAT • DATE • DATETIME
CODED	VARCHAR2(5)	Obsolete.
CODENAME	VARCHAR2(20)	Obsolete.
THESAURUS	VARCHAR2(20)	Name of the coding thesaurus protocol used for coding the item.
LOOKUPNAME	VARCHAR2(20)	Name of the checklist attached to the item.
DERIVED	VARCHAR2(5)	0 — The item is not derived. 1 — The item is derived.
MANDATORY	VARCHAR2(5)	0 — The item is not required. 1 — The item is required.
RANGELB	VARCHAR2(40)	Minimum value for the item.
RANGEUB	VARCHAR2(40)	Maximum value for the item.
DBFMT	VARCHAR2(20)	Database format of the item.
OUTFMT	VARCHAR2(20)	Obsolete.
IORDER	NUMBER(10)	Number of the column containing the item in the update, data, and audit tables for the panel. This value indicates the logical order of items.
PCTNULL	NUMBER(10)	Obsolete.

Column name:	Data type:	Description:
REVORDER	NUMBER(10)	Revised value of IORDER for the item; this value is present only until revisions to the panel are implemented.
CONTEXT_TYPE	NUMBER(2)	0 — Not a context item. 1 — Patient-related context item. 2 — Block-related context item. 3 — Page-related context item. 4 — Other context item.
KEY_ORDER	NUMBER(10)	If the item is part of a user-defined panel key, the order within the key.
MODDATE	DATE	Date and time at which the item was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the item.
DB_ID	NUMBER(5)	Obsolete.
COMPONENT-_ID	NUMBER(10)	Obsolete.
LOCK_STATUS	NUMBER(1)	0 — The item is modifiable. 1 — The item is not modifiable, but it can be reset to modifiable. 2 — The item is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	0 — The item must be copied when the panel is copied. 1 — The item can optionally be copied when the panel is copied.
STATUS	NUMBER(1)	Status of the item: 0 — Valid -1 — Invalid

Column name:	Data type:	Description:
SORT_ORDER	NUMBER(10)	Item used for sorting within observations, and for order within the sort.
SORT_DESC	NUMBER(1)	0 — Sort is descending. 1 — Sort is ascending. This value is null if the item is not used for sorting.
SORT_IS_GRP	NUMBER(1)	0 — Item is not used for sorting or is not in a Type 0 panel. 1 — Items is used for grouping of observations. This value is null if the item is not used for sorting or is not in a Type 0 panel.

Indexes

Index name:	Unique?:	Columns indexed:
ITEM_NONDD_IDX	No	PROTOCOL, UNIQUE_NAME
ITEM_NONDD_PK	Yes	PROTOCOL, PANEL, UNIQUE_NAME

OBJECT_AUDIT

The CTSDD.OBJECT_AUDIT table stores information about modified connected objects.

Rows

One for each modification of a connected source object that has not been used to refresh the destination object.

Columns

Column name:	Data type:	Description:
AUDIT_ID	NUMBER(15)	Unique internal identifier for this record.
PROTOCOL	VARCHAR2(20)	Name of the protocol in which the changed object exists.
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 through 6 — Not applicable. 7 — Study book 8 — Block 9 — Study page 10 — Page layout 11 — Pane (that is, page section) 12 — Panel 13 — Item 14 — Derivation 15 — Rule 16 — Coding target 17 — Thesaurus language 18 — Thesaurus view 19 — Thesaurus algorithm
OBJECT_NAME	VARCHAR2(20)	Name of the connected object that changed.
OBJECT_CONTAINER	VARCHAR2(20)	Name of the container object, if changed object was a contained object.
OBJECT_SUBCONTAINER	VARCHAR2(20)	Name of a contained object, if the contained object is the object that was changed. For page objects, the name of the containing block.
TRANSACTION_TYPE	VARCHAR2(6)	Type of change (UPDATE, DELETE, INSERT).

Column name:	Data type:	Description:
PENDING	NUMBER(1)	0 — Modification is not pending. 1 — Modification was made to an object contained in a panel that is uninstalled or marked for revision.
MODUSER	VARCHAR2(20)	User account that modified the object.
MODDATE	DATE	Date of the modification of the object.
REASON_FOR-_CHANGE	VARCHAR2(255)	User-supplied reason for the change.
DESCRIPTION-_OF_CHANGE	VARCHAR2(255)	User-supplied description of the change.
COMMENTS	VARCHAR2(255)	User-supplied comments.
SUPPORTING-_DOCUMENT	VARCHAR2(255)	User-supplied description of supporting documents.
RELEASE_NUM	NUMBER(15)	Release number of Multisite. If the protocol is in distribution, the distribution release number is in effect at the time of the change.

Index

Index name:	Unique?:	Columns indexed:
OBJECT_AUDIT_PK	Yes	AUDIT_ID

OBJECT_CONNECTION

The CTSDD.OBJECT_CONNECTION table stores information about the connections between source objects and destination objects.

Rows

One for each connected source and destination object.

Columns

Column name:	Data type:	Description:
CONNECTION-_ID	NUMBER(15)	Unique internal identifier for the connection.
SRC_PROTOCOL	VARCHAR2(20)	Name of the protocol from which the object is being copied with a connection.
SRC_OBJECT-_NAME	VARCHAR2(20)	Name of the source object.
SRC_CON-TAINER	VARCHAR2(20)	Name of the source object's container, if it is a contained object.
SRC_SUBCON-TAINER	VARCHAR2(20)	For page objects, the name of the containing block.
DEST_PROTO-COL	VARCHAR2(20)	Name of the protocol to which the object is being copied.
DEST_OBJECT-_NAME	VARCHAR2(20)	Name of the destination object that is being created by a copy with a connection.
DEST_CON-TAINER	VARCHAR2(20)	Name of the destination object's container, if it is a contained object.
DEST_SUBCON-TAINER	VARCHAR2(20)	For page objects, the name of the destination block.

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 through 6 — Not applicable. 7 — Study book 8 — Block 9 — Study page 10 — Page layout 11 — Pane (that is, page section) 12 — Panel 13 — Item 14 — Derivation 15 — Rule 16 — Coding target 17 — Thesaurus language 18 — Thesaurus view 19 — Thesaurus algorithm
COPY_DATE	DATE	Date that the connected object was copied.
DEST_MODDATE	DATE	Obsolete.
PENDING_DEST-_MODDATE	DATE	Date that a connected object was refreshed in the destination instance in a panel that was uninstalled or marked for revision. If Null, there are no changes pending.
SRC_MODIFIED	NUMBER(1)	0 — The source object has not been modified. 1 — The source object has been modified.
SRC_MODIFIED-_PENDING	NUMBER(1)	0 — Modification is not pending. 1 — Modification was made to an object contained in a panel that is uninstalled or marked for revision.

Column name:	Data type:	Description:
REFRESHING	NUMBER(1)	0 — The destination object is not being refreshed. 1 — The destination object is being refreshed.

Indexes

Index name:	Unique?:	Columns indexed:
OBJECT_CONNECTION	Yes	CONNECTION_ID
OBJECT_CONNEC-TION_IDX	No	SRC_PROTOCOL, SRC_OBJECT-_NAME

OBJINDX

The CTSDD.OBJINDX table stores information about Clintrial software text objects. Each object is divided into 2 KB pieces.

Rows

One per 2 KB piece of a Clintrial software text object.

Columns

Column name:	Data type:	Description:
OBJECT_ID	NUMBER(15)	Unique identifier of the object.
OBJECT_SEQ	NUMBER(10)	Sequence number of this piece of the text object.

Column name:	Data type:	Description:
OBJECT_TYPE	VARCHAR2(10)	Type of Clintrial software object: • CTRL_FILE • DERIVATION • PANE • QUERY • QUERY_EXT • RULE • VIEW_PANEL • VIEW_PROT
MODDATE	DATE	Date and time at which the object was last modified.
OBJECT_TEXT	VARCHAR2(2000)	ASCII text.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
OBJINDX_PK	Yes	OBJECT, ONAME, OBJECT_ID, OBJECT_SEQ

PAGELAYOUT

The CTSDD.PAGELAYOUT table stores information about page templates.

Rows

One per page template.

Column

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page template.
LAYOUT_NAME	VARCHAR2(20)	Name of the page template.
MODDATE	DATE	Date and time at which the page template was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the page template.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	Description of the page template.
LOCK_STATUS	NUMBER(1)	0 — The page template is modifiable. 1 — The page template is not modifiable, but it can be reset to modifiable. 2 — The page template is a non-modifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete
STATUS	NUMBER(1)	Status of the page template: 0 — Valid 1 — Invalid

Index

Index name:	Unique?:	Columns indexed:
PAGELAYOUT_PK	Yes	PROTOCOL, LAYOUT_NAME

PAGELAYOUT_EVENT

The CTSDD.PAGELAYOUT_EVENT table stores information about data-entry processing procedures attached to page templates or page sections.

Note: For more information about data-entry processing procedures attached to items, see the CTSDD.PANE_ITEM table.

Rows

One per data-entry processing procedure attached to a page template or page section.

Column

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
LAYOUT_NAME	VARCHAR2(20)	Name of the page template.
PANE_USAGE-_SEQ	NUMBER(5)	If the procedure is attached to a page section, number of the page section.
EVENT_TYPE	NUMBER(2)	Type of event: 1 — Page Opened 2 — Page Saved 3 — Page Deleted 4 — Initializing Page Section 5 — Saving Page Section
EVENT_PROC	VARCHAR2(60)	Name of the data-entry processing procedure.

Index

Index name:	Unique?:	Columns indexed:
PAGELAYOUT_EVENT-_PK	Yes	PROTOCOL, LAYOUT_NAME, PANE_USAGE_SEQ, EVENT_TYPE

PAGE_LIST

The CTSSDD.PAGE_LIST table stores information about page lists.

Rows

One per page list.

Columns

Column name:	Data type:	Description:
LIST_ID	NUMBER(10)	Unique identifier of the page list.
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page list.
LIST_NAME	VARCHAR2(20)	Name of the page list.
LIST_TYPE	NUMBER(10)	0 — All pages are included in the page list. 1 — Specified pages are included in the page list. 2 — Dynamic page list.
LIST_CRITERIA	VARCHAR2(2000)	SQL text of the flag restriction or note restriction associated with the page list.

Indexes

Index name:	Unique?:	Columns indexed:
PAGE_LIST_IDX	Yes	PROTOCOL, LIST_NAME
PAGE_LIST_PK	Yes	LIST_ID

PAGE_LIST_MEMBER

The CTSDD.PAGE_LIST_MEMBER table stores information about pages that are included in page lists.

Rows

One per page that is included in a page list.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page list.
BLOCK_KEY	VARCHAR2(40)	Block key value associated with the page.
PAGE_KEY	VARCHAR2(40)	Page key value associated with the page.
LIST_ID	NUMBER(10)	Unique identifier of the page list.
LIST_ORDER	NUMBER10)	Order of the page in the page list.
BLOCK_REPEAT-_KEY	VARCHAR2(40)	Block repeat key value associated with the page.
PAGE_REPEAT-_KEY	VARCHAR2(40)	Page repeat key value associated with the page.

Index

Index name:	Unique?:	Columns indexed:
PAGE_LIST_MEMBER-_PK	Yes	PROTOCOL, BLOCK_KEY, BLOCK-_REPEAT_KEY, PAGE_KEY, PAGE-_REPEAT_KEY, LIST_ID

PAGE_REF

The CTSDD.PAGE_REF table stores information about pages in a study book.

Rows

One per page in a study book.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the study book that contains the page.
STUDYBOOK-_NAME	VARCHAR2(20)	Name of the study book containing the page.
BLOCK_KEY	VARCHAR2(40)	Block key value associated with the page.
PAGE_KEY	VARCHAR2(40)	Page key value associated with the page.
LAYOUT_NAME	VARCHAR2(20)	Name of the page template associated with the page.
PAGE_ORDER	NUMBER(5)	Order of the page in the block.
PAGE_TITLE	VARCHAR2(30)	Title of the page.

Column name:	Data type:	Description:
REPEATS_LIMIT	NUMBER(10)	Maximum number of page repeats allowed; if -1, there is no limit.
PAGE_NUM	VARCHAR2(20)	Page number of the page.
HELP_CONTEXT-_POINT	NUMBER(5)	Context point number for user-defined Help on this page.
MODDATE	DATE	Date and time at which the page was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the page.
COMPONENT_ID	NUMBER(10)	Obsolete.
DB_ID	NUMBER(5)	Obsolete.
HAS_REPEATS	NUMBER(1)	0 — The study page cannot have repeats. 1 — The study page can have repeat pages.
LOCK_STATUS	NUMBER(1)	0 — The study page is modifiable. 1 — The study page is not modifiable, but it can be reset to modifiable. 2 — The study page is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PAGE_REF_PK	Yes	PROTOCOL, STUDYBOOK_NAME, BLOCK_KEY, PAGE_KEY

PAGE_REF_VALUE

The CTSDD.PAGE_REF_VALUE table stores information about other page-related context items for which default values have been defined.

Rows

One per page-related context item for which a default value has been defined.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the study book that contains the page.
STUDYBOOK-_NAME	VARCHAR2(20)	Name of the study book containing the page.
BLOCK_KEY	VARCHAR2(40)	Block key value associated with the page.
PAGE_KEY	VARCHAR2(40)	Page key value associated with the page.
ITEM_NAME	VARCHAR2(20)	Name of the page-related context item associated with the page identified by the page key value.
ITEM_VALUE	VARCHAR2(240)	Value to be used for the item specified by ITEM_NAME.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PAGE_REF_VALUE_PK	Yes	PROTOCOL, STUDYBOOK_NAME, BLOCK_KEY, PAGE_KEY, ITEM_NAME

PAGE_REPEATS

The CTSDD.PAGE_REPEATS table stores information about repeating pages used in study books.

Rows

One per repeating page.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
BLOCK_KEY	VARCHAR2(40)	Block key value.
BLOCK_REPEAT-_KEY	VARCHAR2(40)	Block repeat key value.
PAGE_KEY	VARCHAR2(40)	Page key value.
PAGE_REPEAT-_KEY	VARCHAR2(40)	Page repeat key value.
PAGE_REPEAT-_ORDER	VARCHAR2(5)	Order of the repeating page within the series of repeating pages. For example, if the page repeats three times, this value could be 1, 2, or 3.
IS_STATIC	NUMBER(1)	0 — The page repeat key was created during data entry. 1 — The page repeat key was predefined by the study designer.
MODDATE	DATE	Date and time at which the repeating page was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the repeating page.

Column name:	Data type:	Description:
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PAGE_RPT_PK	Yes	PROTOCOL, BLOCK_KEY, BLOCK_REPEAT_KEY, PAGE_KEY, PAGE_REPEAT_KEY

PANE

The CTSDD.PANE table stores information about page sections.

Rows

One per page section.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page section.
PANE_NAME	VARCHAR2(20)	Name of the page section.
PANEL_NAME	VARCHAR2(20)	Name of the panel associated with the page section.
HAS_REPEATS	NUMBER(10)	0 — The page section does not contain repeats. 1 — The page section contains repeats.

Column name:	Data type:	Description:
HAS- _SEQUENCES	NUMBER(10)	0 — There are no sequences associated with the page section. 1 — There are sequences associated with the page section.
MAX_REPEATS	NUMBER(10)	Maximum number of repeats allowed in the page section. NULL or -1 means unlimited.
WIDTH	NUMBER(10)	Width of the page section.
HEIGHT	NUMBER(10)	Height of the page section.
STATUS	NUMBER(2)	-2 — No page section layout -1 — Invalid page section layout 0 — Valid page section layout 2 — An object that the page section depends on has changed. The page section layout is still valid, but should be reviewed by the designer.
MODDATE	DATE	Date and time at which the page section was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the page section.
COMPONENT_ID	NUMBER(10)	Obsolete.
OBJECT_ID	NUMBER(15)	Unique identifier of the page section layout description stored in CTSDD- .OBJINDX.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	User-supplied description.

Column name:	Data type:	Description:
LOCK_STATUS	NUMBER(1)	0 — The page section is modifiable. 1 — The page section is not modifiable, but it can be reset to modifiable. 2 — The page section is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PANE_PK	Yes	PROTOCOL, PANE_NAME

PANE_ITEM

The CTSDD.PANE_ITEM table stores information about items in page sections.

Rows

One per item in a page section.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page section that contains the item.
PANE_NAME	VARCHAR2(20)	Name of the page section containing the item.
ITEM_NAME	VARCHAR2(20)	Name of the item.

Column name:	Data type:	Description:
ITEM_STYLE	NUMBER(3)	Display style of the item: 1 — Normal (editable field) 2 — Autoskip 3 — Checklist, drop-down list 4 — Codelist, drop-down list 5 — Check box 6 — Radio button
UC	NUMBER(1)	0 — The case of values entered for the item is ignored. 1 — Values entered for the item must be uppercase.
DUP	NUMBER(1)	0 — The item is not part of an autoduplication group. 1 — The item is part of an autoduplication group.
CARRY	NUMBER(1)	0 — A value for the item is not carried forward from the previous study page in the block. 1 — A value for the item is carried forward from the previous study page in the block.
VERIFY	NUMBER(1)	0 — Verification of the item is not required. 1 — Verification is required before validation can occur.
SEQ_ALIGN	NUMBER(1)	0 — A crossed sequence is attached to the item. 1 — An aligned sequence is attached to the item.
SEQ_CODELIST	VARCHAR2(20)	Name of the codelist used to create initial values for the sequence. Null if a codelist or checklist is attached to the item.

Column name:	Data type:	Description:
HELP	VARCHAR2(240)	Help text that displays at the bottom of the study page when the cursor is in the field for the item.
CONV_PROC	VARCHAR2(60)	Name of the data-entry processing procedure (for the Value Changed event) attached to the item.
OVERRIDE	NUMBER(1)	0 — The upper and lower bounds, and the checklist attached to the item, cannot be overridden. 1 — The upper and lower bounds, and the checklist attached to the item, can be overridden.
CODEENTRY	NUMBER(1)	0 — Values (but not codes) from the attached codelist or checklist can be entered. 1 — Codes (but not values) from the attached codelist can be entered.
CODE_LABEL	VARCHAR2(20)	Reserved.
CODELIST	VARCHAR2(20)	Obsolete.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PANE_ITEM_PK	Yes	PROTOCOL, PANE_NAME, ITEM_NAME

PANE_ITEM_SEQ

The CTSDD.PANE_ITEM_SEQ table stores information about sequences associated with items.

Rows

One per sequence value.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page section that contains the item with a sequence.
PANE_NAME	VARCHAR2(20)	Name of the page section containing an item with a sequence.
ITEM_NAME	VARCHAR2(20)	Name of the item with the sequence.
SEQ_ORDER	NUMBER(6)	Number indicating the order of the sequence value within the sequence. Null if a codelist or checklist is attached to the item.
SEQ_VALUE	VARCHAR2(240)	Sequence value.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PANE_ITEM_SEQ_PK	Yes	PROTOCOL, PANE_NAME, ITEM_NAME, SEQ_ORDER

PANEL

The CTSDD.PANEL table stores information about panels that are installed.

Rows

One per panel that is installed.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel.
PANEL	VARCHAR2(20)	Name of the panel.
PDDNUM	NUMBER(15)	Obsolete.
TYPE	NUMBER(1)	Type of panel (0 – 5).
DESCRIP	VARCHAR2(240)	Description of the panel.
VERIFIABLE	NUMBER(1)	0 — Verification is not required before data in the panel can be validated. 1 — Verification is required before data in the panel can be validated.
INSTALLED	NUMBER(1)	0 — The panel is not yet installed or is deinstalled. 1 — The panel is installed.
TABLES- _CREATED	NUMBER(1)	0 — Database tables for the panel have not been created. 1 — Database tables for the panel have been created.
REVISE_FLAG	NUMBER(1)	0 — The panel is not marked for revision. 1 — The panel is marked for revision.
REVISING	NUMBER(1)	0 — Panel revisions are not currently being implemented. 1 — Panel revisions are currently being implemented.

Column name:	Data type:	Description:
LOCKED	NUMBER(1)	Obsolete.
LOCKUSER	VARCHAR2(20)	Obsolete.
LOCKDATE	DATE	Obsolete.
AUDIT_START	VARCHAR2(20)	Audit start point for the panel: • ENTRY • VERIFICATION • VALIDATION • VALIDITY • MERGE
PROTECTED	NUMBER(1)	0 — The panel is not protected. 1 — The panel is protected.
VLD_ORDER	NUMBER(5)	Order in which the panel should be validated, with respect to other panels.
MODDATE	DATE	Date and time at which the panel was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the panel.
COMPONENT_ID	NUMBER(10)	Obsolete.
VIEW_OBJECT_ID	NUMBER(15)	For a view panel, identifier of the text object that contains the view restriction clause for the panel (if different than the view restriction clause for the protocol) in CTSDD.OBJINDX.
DB_ID	NUMBER(5)	Obsolete.
LOCK_STATUS	NUMBER(1)	0 — The panel is modifiable. 1 — The panel is not modifiable, but it can be reset to modifiable. 2 — The panel is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.

Column name:	Data type:	Description:
LOCK_COPY	NUMBER(1)	Obsolete.
SASNAME	VARCHAR2(8)	SAS name of the panel.
SUBSET_ITEM	VARCHAR2(20)	Name of the panel's subset key item.

Index

Index name:	Unique?:	Columns indexed:
PANEL_PK	Yes	PROTOCOL, PANEL

PANEL_MASTER

The CTSDD.PANEL_MASTER table stores information about panels that are installed and have been defined as detail panels in a master-detail relationship with other panels.

Rows

One per detail panel that has been installed.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
PANEL	VARCHAR2(20)	Name of the detail panel.
ITEM_NAME	VARCHAR2(20)	Name of the detail key item.
MASTER_PANEL	VARCHAR2(20)	Name of the master panel.

Column name:	Data type:	Description:
MASTER_ITEM	VARCHAR2(20)	Name of the master key item.

Index

Index name:	Unique?:	Columns indexed:
PANEL_MASTER_PK	Yes	PROTOCOL, PANEL

PANEL_MASTER_NONDD

The CTSDD.PANEL_MASTER_NONDD table stores information about panels that are not yet installed, are deinstalled, or are marked for revision, and have been defined as the detail panel in a master-detail relationship.

Rows

One per detail panel that is not yet installed, is deinstalled, or is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
PANEL	VARCHAR2(20)	Name of the detail panel.
ITEM_NAME	VARCHAR2(20)	Name of the detail key item.
MASTER_PANEL	VARCHAR2(20)	Name of the master panel.
MASTER_ITEM	VARCHAR2(20)	Name of the master key item.

Index

Index name:	Unique?:	Columns indexed:
PANEL_MASTER-_NONDD_PK	Yes	PROTOCOL, PANEL

PANEL_NONDD

THE CTSDD.PANEL_NONDD table stores information about panels that are not yet installed, are deinstalled, or are marked for revision.

Rows

One per panel that is installed, is deinstalled, or is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel.
PANEL	VARCHAR2(20)	Name of the panel.
TYPE	NUMBER(1)	Type of panel (0 – 5).
DESCRIP	VARCHAR2(240)	Description of the panel.
VERIFIABLE	NUMBER(1)	0 — Verification is not required before data in the panel can be validated. 1 — Verification is required before data in the panel can be validated.
INSTALLED	NUMBER(1)	0 — The panel is not yet installed or is deinstalled. 1 — The panel is installed.

Column name:	Data type:	Description:
TABLES- _CREATED	NUMBER(1)	0 — Database tables for the panel have not been created. 1 — Database tables for the panel have been created.
REVISE_FLAG	NUMBER(1)	0 — The panel is not marked for revision. 1 — The panel is marked for revision.
REVISING	NUMBER(1)	0 — Panel revisions are not currently being implemented. 1 — Panel revisions are currently being implemented.
AUDIT_START	VARCHAR2(20)	Audit start point for the panel: • ENTRY • VERIFICATION • VALIDATION • VALIDITY • MERGE
PROTECTED	NUMBER(1)	0 — The panel is not protected. 1 — The panel is protected.
VLD_ORDER	NUMBER(5)	Order in which the panel should be validated, with respect to other panels.
MODDATE	DATE	Date and time at which the panel was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the panel.
VIEW_OBJECT_ID	NUMBER(15)	For a view panel, identifier of the text object that contains the view restriction clause for the panel (if different than the view restriction clause for the protocol) in CTSDD.OBJINDX.

Column name:	Data type:	Description:
LOCK_STATUS	NUMBER(1)	0 — The panel is modifiable. 1 — The panel is not modifiable, but it can be reset to modifiable. 2 — The panel is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
SASNAME	VARCHAR2(8)	SAS name of the panel.
SUBSET_ITEM	VARCHAR2(20)	Name of the panel's subset key item.

Index

Index name:	Unique?:	Columns indexed:
PANE_NONDD_PK	Yes	PROTOCOL, PANEL

PANE_SEQ_VALUE

THE CTSDD.PANE_SEQ_VALUE table stores information about sequences associated with items, after the sequences have been crossed or aligned with other sequences in the page section.

Rows

One per sequence value.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the page section that contains the item with a sequence.

Column name:	Data type:	Description:
PANE_NAME	VARCHAR2(20)	Name of the page section containing an item with a sequence.
ITEM_NAME	VARCHAR2(20)	Name of the item with the sequence.
SEQ_ORDER	NUMBER(6)	Number indicating the order of the sequence value.
SEQ_VALUE	VARCHAR2(240)	Sequence value.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
PANE_SEQ_VALUE_PK	Yes	PROTOCOL, PANE_NAME, ITEM_NAME, SEQ_ORDER

PANE_USAGE

The CTSDD.PANE_USAGE table stores information about page sections that are used in page templates.

Rows

One per use of a page section.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol in which the page section is used.

Column name:	Data type:	Description:
LAYOUT_NAME	VARCHAR2(20)	Name of the page template that uses the page section.
PANE_USAGE-_SEQ	NUMBER(5)	Number associated with the page section within the page template. This number does not determine layout or tabbing order, and does not change when layout or tabbing order changes.
PANE_NAME	VARCHAR2(20)	Name of the page section.
PANE_ORDER	NUMBER(10)	Tabbing order of the page section in the page template.
POSITION_X	NUMBER(10)	Vertical position of the page section.
POSITION_Y	NUMBER(10)	Horizontal position of the page section.
DB_ID	NUMBER(5)	Obsolete.
SUBSET_VALUE	VARCHAR2(40)	Value of the subset key item for the page section.

Index

Index name:	Unique?:	Columns indexed:
PANE_USAGE_PK	Yes	PROTOCOL, LAYOUT_NAME, PANE_USAGE_SEQ

QUERY

The CTSDD.QUERY table stores information about queries created in Retrieve.

Rows

One per query.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the query.
QUERY_NAME	VARCHAR2(20)	Name of the query.
DESCRIP	VARCHAR2(80)	Description of the query.
QUERY_TYPE	NUMBER(10)	1 — QBF 2 — Ad Hoc 3 — QBP 4 — SQL
IS_SYSTEM	NUMBER(1)	0 — Private query 1 — Public query
EXEC_DATE	DATE	Date and time at which the query was last run.
STATUS	NUMBER(2)	0 — Created and not modified. 1 — Modified.
MODDATE	DATE	Date and time at which the query was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the query.
OBJECT_ID	NUMBER(15)	Identifier of the text object that contains the query text in CTSDD.OBJINDX.
EXT_OBJECT_ID	NUMBER(15)	Reserved
DB_ID	NUMBER(5)	Obsolete
SUMMARY_INFO	VARCHAR2(255)	Summary description of the query, for use with Integrated Review™.

Index

Index name:	Unique?:	Columns indexed:
QUERY_PK	Yes	PROTOCOL, QUERY_NAME

RULE

The CTSDD.RULE table stores information about rules.

Rows

One per rule.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the panel to which the rule is attached.
PANEL	VARCHAR2(20)	Name of the panel to which the rule is attached.
RULE_NAME	VARCHAR2(20)	Name of the rule.
RULE_ACTION	VARCHAR2(10)	Rule action of the rule: • REPORT • REJECT
IS_COMPILED	NUMBER(1)	0 — The rule has not been compiled successfully. 1 — The rule has been compiled successfully.

Column name:	Data type:	Description:
MSG_IS- _DERIVED	NUMBER(1)	0 — The message is not created by a derivation. 1 — The message is created by a derivation.
MSG_TEXT	VARCHAR2(240)	Text of the message generated by the rule if the record fails the rule.
EMPTY_IS_TRUE	NUMBER(1)	0 — If the rule evaluates to EMPTY, it is treated as if it evaluated to FALSE. 1 — If the rule evaluates to EMPTY, it is treated as if it evaluated to TRUE.
TAGID	NUMBER(15)	Unique identifier associated with the tag that is set if the rule evaluates to FALSE.
MODDATE	DATE	Date and time at which the rule was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the rule.
OBJECT_ID	NUMBER(15)	Identifier of the text object that contains the text of the rule in CTSDD.OBJINDX.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	User-supplied description.
LOCK_STATUS	NUMBER(1)	0 — The rule is modifiable. 1 — The rule is not modifiable, but it can be reset to modifiable. 2 — The rule is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	0 — The rule must be copied when the panel is copied. 1 — The rule can optionally be copied when the panel is copied.

Column name:	Data type:	Description:
PRIORITY	NUMBER(5)	Priority of discrepancies generated by the rule.
DISCREP_STATE	VARCHAR2(8)	Initial status of discrepancies generated by the rule.

Index

Index name:	Unique?:	Columns indexed:
RULE_PK	Yes	PROTOCOL, PANEL, RULE_NAME

RULE_AUDIT

The CTSDD.RULE_AUDIT table stores information about modified or deleted rules while the panel is marked for revision.

Rows

One per modified or deleted rule while the panel is marked for revision.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Same value as that of the rule before it was modified or deleted.
PANEL	VARCHAR2(20)	Same value as that of the rule before it was modified or deleted.
RULE_NAME	VARCHAR2(20)	Same value as that of the rule before it was modified or deleted.

Column name:	Data type:	Description:
RULE_ACTION	VARCHAR2(10)	Same value as that of the rule before it was modified or deleted.
IS_COMPILED	NUMBER(1)	Same value as that of the rule before it was modified or deleted.
MSG_IS-_DERIVED	NUMBER(1)	Same value as that of the rule before it was modified or deleted.
MSG_TEXT	VARCHAR2(240)	Same value as that of the rule before it was modified or deleted.
EMPTY_IS_TRUE	NUMBER(1)	Same value as that of the rule before it was modified or deleted.
TAGID	NUMBER(15)	Same value as that of the rule before it was modified or deleted.
MODDATE	DATE	Same value as that of the rule before it was modified or deleted.
MODUSER	VARCHAR2(20)	Same value as that of the rule before it was modified or deleted.
OBJECT_ID	NUMBER(15)	Same value as that of the rule before it was modified or deleted.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	Same value as that of the rule before it was modified or deleted.
LOCK_STATUS	NUMBER(1)	Same value as that of the rule before it was modified or deleted.
LOCK_COPY	NUMBER(1)	Same value as that of the rule before it was modified or deleted.
PRIORITY	NUMBER(5)	Same value as that of the rule before it was modified or deleted.
DISCREP_STATE	VARCHAR2(8)	Same value as that of the rule before it was modified or deleted.

Index

Index name:	Unique?:	Columns indexed:
RULE_AUDIT_PK	Yes	PROTOCOL, PANEL, RULE_NAME

STUDYBOOK

The CTSDD.STUDYBOOK table stores information about study books.

Rows

One per study book.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the study book.
STUDYBOOK-_NAME	VARCHAR2(20)	Name of the study book.
DESCRIPTION	VARCHAR2(240)	Description of the study book.
CLASS	NUMBER(10)	Type of panels associated with the study book: 0 — A single Type 0 panel. 1 — Type 1–4 panels. 5 — A single Type 5 panel.
STATUS	NUMBER(1)	Status of the study book: 0 — Valid 1 —Invalid

Column name:	Data type:	Description:
MODDATE	DATE	Date and time at which the study book was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the study book.
COMPONENT-_ID	NUMBER(10)	Obsolete.
DB_ID	NUMBER(5)	Obsolete.
LOCK_STATUS	NUMBER(1)	0 — The study book is modifiable. 1 — The study book is not modifiable, but it can be reset to modifiable. 2 — The study book is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
STUDYBOOK_PK	Yes	PROTOCOL, STUDYBOOK_NAME

SUBJECT_LIST

The CTSDD.SUBJECT_LIST table stores information about subject lists.

Rows

One per subject list.

Columns

Column name:	Data type:	Description:
LIST_ID	NUMBER(10)	Unique identifier of the subject list.
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the subject list.
LIST_NAME	VARCHAR2(20)	Name of the subject list.
LIST_TYPE	NUMBER(10)	0 — All subjects are included in the subject list. 1 — Specified subjects are included in the subject list. 2 — Dynamic subject list. 3 — Clintrace subject list. 4 — Site list, which is automatically defined when the protocol is registered for replication.
LIST_CRITERIA	VARCHAR2(2000)	SQL text of the tag restriction associated with the subject list.

Index

Index name:	Unique?:	Columns indexed:
SUBJECT_LIST_IDX	Yes	PROTOCOL, LIST_NAME
SUBJECT_LIST_PK	Yes	LIST_ID

SUBJECT_LIST_MEMBER

The CTSDD.SUBJECT_LIST_MEMBER table stores information about subjects that are included in subject lists.

Rows

One per subject included in a subject list.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the subject list.
SUBJECT_ID	NUMBER(15)	Identifier of the subject in the subject list.
LIST_ID	NUMBER(10)	Unique identifier of the subject list.
LIST_ORDER	NUMBER(10)	Order of the subject in the subject list.

Index

Index name:	Unique?:	Columns indexed:
SUBJ_LIST- _MEMBER_PK	Yes	PROTOCOL, SUBJECT_ID, LIST_ID

THESAURUS_ALGORITHM

The CTSDD.THESAURUS_ALGORITHM table stores information about customized thesaurus algorithms used for coding.

Rows

One per customized thesaurus algorithm.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the customized thesaurus algorithm.
ALGORITHM	VARCHAR2(20)	Name of the customized thesaurus algorithm.
MODDATE	DATE	Date and time at which the customized thesaurus algorithm was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the customized thesaurus algorithm.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	Description of the algorithm.
NORMALIZE	NUMBER(1)	0 — Comprehensive normalization does not occur before use of this thesaurus algorithm. 1 — Comprehensive normalization occurs before use of this thesaurus algorithm.
LOCK_STATUS	NUMBER(1)	0 — The thesaurus algorithm is modifiable. 1 — The thesaurus algorithm is not modifiable, but it can be reset to modifiable. 2 — The thesaurus algorithm is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.
PREF_TERM- _VIEW	VARCHAR2(20)	Thesaurus view used to define the preferred terms.

Index

Index name:	Unique?:	Columns indexed:
THESAURUS- _ALGORITHM_PK	Yes	PROTOCOL, ALGORITHM

THESAURUS_ALGORITHM_STEP

The CTSDD.THESAURUS_ALGORITHM_STEP table stores information about the steps of customized thesaurus algorithms.

Rows

One per step of a customized thesaurus algorithm.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the customized thesaurus algorithm.
ALGORITHM	VARCHAR2(20)	Name of the customized thesaurus algorithm.
STEP_ORDER	NUMBER(5)	Number of the algorithm step.
STEP_TYPE	VARCHAR2(20)	Type of activity performed by the algorithm step: • EXACT • CONTAINS • FILTER If Classify is loaded, additional step-type activities are possible.
VIEW_NAME	VARCHAR2(20)	Name of the thesaurus view that is used by the algorithm step.

Column name:	Data type:	Description:
MODDATE	DATE	Date and time at which the algorithm step was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the algorithm step.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
THESAURUS-_ALGORITHM_STEP_PK	Yes	PROTOCOL, ALGORITHM, STEP_ORDER

THESAURUS_LANGUAGE

The CTSDD.THESAURUS_LANGUAGE table stores information about thesaurus languages used for coding.

Rows

One per thesaurus language.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol containing the thesaurus language.
LANGUAGE	VARCHAR2(20)	Name of the thesaurus language.

Column name:	Data type:	Description:
PUNCTUATION	VARCHAR2(100)	Punctuation marks removed from verbatim text during automatic coding using the thesaurus language.
MODDATE	DATE	Date and time at which the thesaurus language was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the thesaurus language.
DB_ID	NUMBER(5)	Obsolete.
LOCK_STATUS	NUMBER(1)	0 — The thesaurus language is modifiable. 1 — The thesaurus language is not modifiable, but it can be reset to modifiable. 2 — The thesaurus language is a nonmodifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
THESAURUS_LANG_PK	Yes	PROTOCOL, LANGUAGE

THESAURUS_VIEW

The CTSDD.THESAURUS_VIEW table stores information about thesaurus views used for coding.

Rows

One per thesaurus view.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the thesaurus protocol containing the thesaurus view.
VIEW_NAME	VARCHAR2(20)	Name of the thesaurus view. The default thesaurus algorithm requires creation of the following views: • TERMS • SYNONYMS • STOPWORDS
PANEL_OWNER	VARCHAR2(20)	For views based on tables, the user account that owns the table. Null for views based on panels in the coding thesaurus protocol.
PANEL	VARCHAR2(20)	For views based on tables, name of the base Oracle table. For views based on panels, name of the base panel in the coding thesaurus protocol.
CODE1-_COLUMN	VARCHAR2(20)	Name of the item that contains the code (or the first part of a multipart code).
CODE2-_COLUMN	VARCHAR2(20)	Name of the item that contains the second part of a multipart code.
CODE3-_COLUMN	VARCHAR2(20)	Name of the item that contains the third part of a multipart code.
TEXT_COLUMN	VARCHAR2(20)	Name of the item that contains the text.
ACTIVE-_COLUMN	VARCHAR2(20)	Name of the item that indicates whether the record is active (can be used for coding) in the thesaurus view.
ACTIVE_VALUE	VARCHAR2(10)	Value of the ACTIVE_COLUMN that indicates Active.
VIEWS-_CREATED	NUMBER(1)	0 — Views have not been created. 1 — Views have been created.

Column name:	Data type:	Description:
MODDATE	DATE	Date and time at which the thesaurus view was created or last modified.
MODUSER	VARCHAR2(20)	User account that created or last modified the thesaurus view.
DB_ID	NUMBER(5)	Obsolete.
DESCRIPTION	VARCHAR2(240)	Description of the thesaurus view.
LOCK_STATUS	NUMBER(1)	0 — The thesaurus view is modifiable. 1 — The thesaurus view is not modifiable, but it can be reset to modifiable. 2 — The thesaurus view is a non-modifiable copy that cannot be made modifiable except by breaking the connection.
LOCK_COPY	NUMBER(1)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
THESAURUS_VIEW_PK	Yes	PROTOCOL, VIEW_NAME

8 CTSRM Account

Overview

The CTSRM account stores information for Multisite Distribution.

CODE_VALUE_DIFF

The CTSRM.CODE_VALUE_DIFF table stores information about differences in codelist values. Differences in codelist values for aggregated and unaggregated codelists are stored in this table.

A row is created in this table when a distributed codelist is closed for revision and its value has been modified or a new value has been created.

Rows

One for each value inserted or deleted; two for each value changed.

Columns

Column name:	Data type:	Description:
CODELIST	VARCHAR2(20)	Name of the codelist.
CODE	VARCHAR2(80)	Codelist code.
VALUE	VARCHAR2(80)	Codelist value.
NEW_VERSION	NUMBER(1)	0 — Old or deleted. 1 — New or inserted.
LABEL	VARCHAR2(80)	Codelist label.
LONGLABEL	VARCHAR2(240)	Codelist long label.
STATUS	NUMBER(1)	Status of the codelist value.

Column name:	Data type:	Description:
CODE_ORDER	NUMBER(5)	Order in codelist.
SUBSET_REQD	NUMBER(1)	0 — Value not required in all codelists. 1 — Value required in all codelists.
SUBSET_VALUE	NUMBER(5)	Value to distinguish subsets.

DIFF_ANALYSIS

The CTSRM.DIFF_ANALYSIS table stores information about modifications made to distributed protocols.

A row is created in this table when a protocol is closed for revision after modifications have been made to an object in the protocol.

Rows

One for each changed object.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the distributed protocol.

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 — Protocol 3 — Codelist 2, 4 through 6 — Reserved 7 — Study book 8 — Block 9 — Study page 10 — Page layout 11 — Page Section 12 — Panel 13 — Item 14 — Derivation 15 — Rule 16 — Coding target 17 — Thesaurus language 18 — Thesaurus view 19 — Thesaurus algorithm 20 — Query 100 — Lab Loader Map 101 — Lab Loader Control File
OBJECT_NAME	VARCHAR2(30)	Name of the modified object.
OBJECT- _CONTAINER	VARCHAR2(30)	If the object that changed is a contained object, the name of the container object for that contained object (for example, the name of the panel that contains items).
OBJECT_SUB- _CONTAINER	VARCHAR2(30)	Name of subcontainer of the modified object.

Column name:	Data type:	Description:
CHANGE_TYPE	VARCHAR2(6)	Type of change to the object: • INSERT • UPDATE • DELETE

Index

Index name:	Unique?:	Columns indexed:
DIFF_ANALYSIS_IDX	No	PROTOCOL, OBJECT_TYPE, OBJECT_NAME, OBJECT_CONTAINER, OBJECT_SUBCONTAINER

ERRORLOG

The CTSRM.ERRORLOG table stores information about errors that have occurred as part of the distribution process.

A row is created in this table when a site attempts to accept an object that has errors at the site.

Rows

One per error of each object where acceptance failed at the site.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the object with the error.
RELEASE_NUM	NUMBER(15)	Internal release number of the object with the error.
TABLE_NAME	VARCHAR2(30)	Table name where the error occurred.
ERRDT	DATE	Date the error was detected.
ERRACT	VARCHAR2(20)	The action associated with the error: • REJECT • REPORT
REMARKS	VARCHAR2(240)	Informational text about the error.

Index

Index name:	Unique?:	Columns indexed:
ERRORLOG_IDX	No	OBJECT_TYPE, OBJECT_NAME, RELEASE_NUM

FUNCTION_RECV

The CTSRM.FUNCTION_RECV table stores information about PL/SQL functions, procedures, and packages received at a site.

A row is created in this table when a function, procedure, or package is copied to a site.

Rows

One for each package, procedure, or function received.

Columns

Column name:	Data type:	Description:
SRC_INSTANCE	VARCHAR2(30)	Name of the site where the function originated.
OWNER	VARCHAR2(30)	Account name that owns the function.
OBJECT_NAME	VARCHAR2(30)	Name of the function, package or procedure.
OBJECT_TYPE	VARCHAR2(15)	Type of object: • FUNCTION • PACKAGE • PROCEDURE • PACKAGE BODY
OBJECT_LENGTH	NUMBER	Length of the function.
STATUS	NUMBER(1)	Status of the function: 0 – Pending. 1 – Compiled. 2 – Compile Error.
MODDATE	DATE	Date the function was modified.
MODUSER	VARCHAR2(20)	User who modified the function.

Index

Index name:	Unique?:	Columns indexed:
FUNCTION_RECV_PK	Yes	SRC_INSTANCE, OWNER, OBJECT_NAME, OBJECT_TYPE

FUNCTION_SOURCE

The CTSRM.FUNCTION_SOURCE table stores the source PL/SQL code for functions, procedures, and packages received at a site.

A row is created in this table when a function, procedure, or package is copied to a site.

Rows

One for each line of source code.

Columns

Column name:	Data type:	Description:
SRC_INSTANCE	VARCHAR2(30	Name of the site where the function originated.
OWNER	VARCHAR2(30)	Account name that owns the function.
OBJECT_NAME	VARCHAR2(30)	Name of the function, package or procedure.
OBJECT_TYPE	VARCHAR2(15)	Type of object: • FUNCTION • PACKAGE • PROCEDURE

Column name:	Data type:	Description:
LINE	NUMBER	The line number of the source.
TEXT	VARCHAR2(4000)	The text that comprises this portion of the source.

Index

Index name:	Unique?:	Columns indexed:
FUNCTION_SOURCE_PK	Yes	SRC_INSTANCE, OWNER, OBJECT_NAME, OBJECT_TYPE, LINE

ObjectTable_DIFF

The ObjectTable_DIFF tables store information about differences between current objects and objects as they existed when the object was last compared. Possible values of ObjectTable are as follows.

Protocol Distribution:

BLOCK_REF
BLOCK_REF_VALUE
BLOCK_REPEATS
DERIVATION
ENCODING_TARGET
ITEMOBJINDX
PAGELAYOUT
PAGE_REF
PAGE_REF_VALUE
PAGE_REPEATS
PANE
PANE_ITEM
PANE_ITEM_SEQ
PANEL

PANEL_MASTER
PANE_SEQ_VALUE
PANE_USAGE
RULE
STUDYBOOK
THESAURUS_ALGORITHM
THESAURUS_ALGORITHM_STEP
THESAURUS_LANGUAGE
THESAURUS_VIEW

PROTOCOL
PROTOCOL_PARAM

SEARCH_LIST

QUERY

Codelist Distribution:

CODE_INDEX
AGGREGATED_CODES
VIEW_CODELIST

Rows

One for each insertion or deletion of an object; two for each modification of an existing object.

Columns

For each entry in the previous list, the *object-name*_DIFF table columns are identical to the columns in the tables where the objects are defined, except for the columns noted in the following table (and system items, which the *object-name*_DIFF tables do not contain). For example, the CTSRM.Block_Ref_DIFF table has the same columns as CTSDD.BLOCK_REF.

Column name:	Data type:	Description:
ObjectKeys	Data type(s) of key item(s) for the object.	Column name(s) are the names of the key item(s) for the object. Values are the value(s) of key item(s) for the object.

Column name:	Data type:	Description:
Modifiable-Attributes		Values of the user-modifiable attributes of the object's base table. Attributes maintained by the Clintrial software are *not* included, such as: • MODDATE • MODUSER • COMPONENT_ID • DB_ID • STATUS (for the objects STUDYBOOK, PANE, and ENCODING_TARGET) • OBJECT_ID (for the objects PANE, RULE, and DERIVATION) • IS_COMPILED (for the objects RULE and DERIVATION) • VIEWS_CREATED (for the object THESAURUS_VIEW)
NEW_VERSION	NUMBER(1)	0 — The record contains previous values of object attributes. 1 — The record contains new values of object attributes.

ObjectTable_SN

The ObjectTable_SN tables store copies of the metadata at the time a distributed object is opened for revision. Possible values of ObjectTable are listed in the description of ObjectTable_DIFF. The same comment about keys noted in the ObjectTable_DIFF apply to the ObjectTable_SN table.

Rows

One for each row at the time the object was opened.

Columns

The columns in the SN tables are the same as the columns for the objects themselves. For example the columns in CTSRM.BLOCK_REF_SN are the same as the columns in CTSDD.BLOCK_REF. (The _SN tables do include the columns maintained by the Clintrial software; for example, MODDATE, MODUSER.)

OBJINDX_SN

The OBJINDX_SN table stores values of unlimited attributes at the time a distributed protocol is opened for revision. Unlimited text is used by Rules, Derivations, View Protocols, Queries, and Page Section Layouts.

Rows

One for each block of text.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
OBJECT- _CONTAINER	VARCHAR2(20)	If the changed object is a contained object, the name of the container object for that contained object
OBJECT_NAME	VARCHAR2(20)	Name of the modified object.
OBJECT_SEQ	NUMBER(10)	Text block sequence number within the unlimited text attribute.
OBJECT_TYPE	VARCHAR2(10)	VIEW_PROT, VIEW_PANEL, RULE, DERIVATION, PANE, QUERY, QUERY_EXT
MODDATE	DATE	Date of modification

Column name:	Data type:	Description:
OBJECT_TEXT	VARCHAR2(2000)	Text block.

Index

Index name:	Unique?:	Columns indexed:
OBJINDX_SN_PK	Yes	PROTOCOL, OBJECT_CONTAINER, OBJECT_NAME, OBJECT_SEQ, OBJECT_TYPE

RELEASE_CHANGE

The CTSRM.RELEASE_CHANGE table stores information about changes to metadata distributed to a site or received at a site.

Rows are created in this table at the source when a new release is distributed to another site.

Rows

One for each metadata object which is new or changed.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the object.

Column name:	Data type:	Description:
RELEASE_NUM	NUMBER(15)	Internal numeric identifier for the release.
IS_INITIAL	NUMBER(1)	Is this the initial release for this destination? 0 – No 1 – Yes
TABLE_NAME	VARCHAR2(30)	Table that was modified.
CHANGE_TYPE	VARCHAR2(6)	Type of change (UPDATE, DELETE, INSERT).
APPLIED	NUMBER(1)	0 — The change has been applied. 1 — The change has not been applied.
1 through I24	VARCHAR2(255)	Values of the object attributes; dependent on the object type.
125	VARCHAR2(2000)	Values of the object attributes; dependent on the object type.

Index

Index name:	Unique?:	Columns indexed:
RELEASE_CHANGE-_IDX	No	OBJECT_TYPE, OBJECT_NAME, RELEASE_NUM, IS_INITIAL, TABLE_NAME, CHANGE_TYPE

RELEASE_RECV

The CTSRM.RELEASE_RECV table stores information about each object accepted at the site.

A row is created in this table when an object is accepted at the site.

Rows

One per each accepted object at the site.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was changed: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the distributed object.
RELEASE_NUM	NUMBER(15)	The current release number of the distributed object.
STATUS	NUMBER(1)	Status of the object: 0 — Pending 1 — Accepted 2 — Accept failed
MODDATE	DATE	Date the object was distributed.
MODUSER	VARCHAR2(20)	User who distributed the object.

Index

Index name:	Unique?:	Columns indexed:
RELEASE_SEND_PK	Yes	OBJECT_TYPE, OBJECT_NAME, RELEASE_NUM, DEST_NAME

RELEASE_SEND

The CTSRM.RELEASE_SEND table stores information about each object distributed from the site

A row is created in this table when an object is distributed from the site.

Rows

One per each distributed object per destination site.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of distributed object: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the distributed object.
RELEASE_NUM	NUMBER(15)	Current release number of the distributed object.
DEST_NAME	VARCHAR2(20)	Name of the site to which the object was distributed.
MODDATE	DATE	Date the object was distributed.
MODUSER	VARCHAR2(20)	User who distributed the object.

Index

Index name:	Unique?:	Columns indexed:
RELEASE_SEND_PK	Yes	OBJECT_TYPE, OBJECT_NAME, RELEASE_NUM, DEST_NAME

RELEASE_VERSION

The CTSRM.RELEASE_VERSION table stores information about the version of each object distributed to or from the site.

A row is created in this table when an object is first distributed from the site, or when the object is closed for revision.

Rows

One per each version of each distributed object.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of object that was closed or distributed: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the distributed object.
RELEASE_NUM	NUMBER(15)	Internal release number of the distributed object.

Column name:	Data type:	Description:
VERSION_NUM	VARCHAR2(20)	Version number of the distributed object, entered by the user when the object was distributed or closed for revision.
DESCRIPTION	VARCHAR2(240)	Description of the distributed object, entered by the user when the object was distributed or closed for revision.

Index

Index name:	Unique?:	Columns indexed:
RELEASE_VERSION_PK	Yes	OBJECT_TYPE, OBJECT_NAME, RELEASE_NUM

RELEASED_OBJECT

The CTSRM.RELEASED_OBJECT table stores information about the objects that have been distributed to or from the site.

A row is created in this table when an object is first distributed to or from the site.

Rows

One per distributed object.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of distributed object: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the distributed object.
LOCKED	NUMBER(1)	Status of the distributed object: 0 — Open for revision at the local site. 1 — Closed for revision or an object at a subordinate site.
SRC_INSTANCE	VARCHAR2(30)	If the distributed object was accepted from another site, the name of that Source site.
LAST_COMPARED	DATE	Date of the last comparison between revisions of the object.

Index

Index name:	Unique?:	Columns indexed:
RELEASED- _OBJECT_PK	Yes	OBJECT_TYPE, OBJECT_NAME

LATEST_RECV view

The CTSRM.LATEST_RECV view stores information about most recent versions of objects that have been received at the site. This view is created from the RELEASED_OBJECT, RELEASE_VERSION, and the RELEASE_RECV tables.

Rows

One per distributed object.

Columns

Column name:	Data type:	Description:
OBJECT_TYPE	NUMBER(5)	Type of distributed object: 1 — Protocol 3 — Codelist
OBJECT_NAME	VARCHAR2(20)	Name of the distributed object.
LOCKED	NUMBER(1)	Status of the object: Always 1 – Closed for revision
SRC_INSTANCE	VARCHAR2(30)	If the distributed object was accepted from another site, the name of that Source site.
LAST_COMPARED	DATE	Date of the last comparison between revisions of the object.
RELEASE_NUM	NUMBER(15)	Current release number of the distributed object.
STATUS	NUMBER(1)	Status of the object: 0 — Pending 1 — Accepted 2 — Accept failed
VERSION_NUM	VARCHAR2(20)	Version number of the distributed object, entered by the user when the object was distributed or closed for revision.
DESCRIPTION	VARCHAR2(240)	Description of the distributed object, entered by the user when the object was distributed or closed for revision.

9 CTSRP Account

Overview

The CTSRP account stores information for Multisite Replication.

CALLREC

This table is no longer used.

CHANGEREC

The CTSRP.CHANGEREC table stores information about changes made to records in tables in replication.

A row is created in this table when a change is made to a record in a table which is being replicated.

Rows

One for each record change.

Columns

Column name:	Data type:	Description:
CHANGENUM	NUMBER	Sequence assigned to the change.
CHANGETYPE	CHAR(1)	Type of change: I — Insert U — Update D — Delete L — Local

Column name:	Data type:	Description:
CHANGETAB	VARCHAR2(30)	Table where change occurred.
CHANGEGROUP	VARCHAR2(30)	Account where change occurred.
TOSITE	NUMBER(6)	DB_ID of the site where the change will be replicated.
FROMSITE	NUMBER(6)	DB_ID of the site that made the change.

Index

Index name:	Unique?:	Columns indexed:
PK_CHANGEREC	Yes	CHANGENUM

DCHANGENUM

The CTSRP.DCHANGENUM table stores information about the number of record changes made in a group.

Rows are created in this table when a Replication Master site is ready to replicate data to a Replication Subordinate site.

Rows

One for each change to be replicated to a Replication Subordinate site.

Columns

Column name:	Data type:	Description:
SITEID	NUMBER(6)	DB_ID of the site where the change is made.
GROUPINDX	NUMBER(6)	Index of the account.
CHANGENUM	NUMBER	The change sequence.

Indexes

None.

DBOTYPEINFO

The CTSRP.DBOTYPEINFO table stores information about parameters used in the creation tables and sequences.

Rows

A row is created in this table at installation time.

Columns

Column name:	Data type:	Description:
INDX	NUMBER(5)	

Column name:	Data type:	Description:
TYPENAME	VARCHAR2(20)	One of the following: • DEFAULT • TBL_ADMIN • USR_DEFAULT • TBL_USER At installation, information related to the creation of tables and sequences for replication objects is stored in the N1 through N9 and S1 through S3 columns. The type of information that is stored in those columns depends on whether this type is DEFAULT (information supplied by the Clintrial software) or one of the other five types, for which information is supplied at installation.
PARENT	NUMBER(5)	Not used.
N1	NUMBER	
N2	NUMBER	
N3	NUMBER	
N4	NUMBER	
N5	NUMBER	
N6	NUMBER	
N7	NUMBER	
N8	NUMBER	
N9	NUMBER	
S1	VARCHAR2(40)	
S2	VARCHAR2(40)	
S3	VARCHAR2(40)	

Indexes

None.

ERRORREC

The CTSRP.ERRORREC table stores information about replication errors.

A row is created in this table when a replication error occurs for a table in replication.

Rows

One for each error.

Columns

Column name:	Data type:	Description:
ERRORNUM	NUMBER	Sequence identifying the error.
ERRORTYPE	CHAR(1)	Type of error: • I — Insert • U — Update • D — Delete
ISUPLOAD	CHAR(1)	Did the error occur when uploading data (Replication Subordinate site to Replication Master site)? Y — Yes N — No
CHANGENUM	NUMBER	Change number from CTSRP.CHANGEREC associated with this error.
ERRORTAB	VARCHAR2(30)	Table where the error occurred.

Column name:	Data type:	Description:
ERRORCODE	NUMBER(8)	Oracle error code associated with the error.
ERRORGROUP	VARCHAR2(30)	The account (group) where the error occurred.
ERRORMSG	VARCHAR2 (240)	The error message associated with the error.
ERRORTOSITE	NUMBER(6)	DB_ID of the site where the record is being replicated.

Index

Index name:	Unique?:	Columns indexed:
PK_ERRORREC	Yes	ERRORNUM

GROUPDIST table

The CTSRP.GROUPDIST table stores information about replicated accounts per site.

A row is created in this table when a site is added into replication for the account.

Rows

One for each site that is added into replication.

Columns

Column name:	Data type:	Description:
SITEID	NUMBER(6)	DB_ID of the site particpating in replication.
GROUPINDX	NUMBER(6)	Index of the replicating account.
PRIVLS	CHAR(2)	Privileges of the account: • RW — Read/write • RO — Read only
STATUS	CHAR(2)	Current status of account at the site. See *Multisite* for a list of possible statuses.
STATGOAL	CHAR(2)	Goal Status of account at the site. See *Multisite* for a list of possible statuses.
LASTCHANGE	NUMBER(8)	Sequence identifying the last replication change made for the account to the site.
JOBNO	NUMBER(6)	Oracle job number replicating this account.

Indexes

None.

HSUBVIEW

The CTSRP.HSUBVIEW table stores information about horizontal subsets applied to tables in replication. It is only populated at the Replication Master site of the account.

A row is created in this table when a subset is placed on a replicating table.

Rows

One for each table and site combination.

Columns

Column name:	Data type:	Description:
SITEID	NUMBER(6)	DB_ID of the site where the WHERE clause is applied.
TABLEINDX	NUMBER(6)	Index assigned to the replicating table from CTSRP.REPTABLE.
WCLAUSE	VARCHAR2 (1000)	Current SQL restriction.
OLDWCLAUSE	VARCHAR2 (1000)	Previous SQL restriction.
SUBTYPE	CHAR(2)	Subset type: HZ — Horizontal.
UPDATED	CHAR(1)	Indicates if the WHERE clause (subset) has been updated: • Y — Yes • N — No

Index

Index name:	Unique?:	Columns indexed:
U_SITE_TABLE_TYPE	Yes	SITEID, TABLEINDX, SUBTYPE

REPAUDIT

The CTSRP.REPAUDIT table stores information about the replication history of the table.

A row is created in this table when a replication event occurs.

Rows

One for each replication event per group per site.

Columns

Column name:	Data type:	Description:
EVENT_DATE	DATE	The date the replication event occurred.

Column name:	Data type:	Description:
EVENT_CODE	CHAR(2)	Replication event code associated with the change: UP — upload UD — upload done UE — upload error UR — redo upload errors DW — download begin DD — download done DE — download error DR — redo download errors HG — halt group RG — resume group EC — clear errors EF — initial download HC — halt client HD — download while halting RC — resume client DC — drop client AL — accepting invitation TA — accepted invitation FI — Finishing invitation CH — halting IU — initial uploading CD — dropping See *Multisite* for descriptions of the event codes.
CDATA1	VARCHAR2 (100)	Name of a protocol.
CDATA2	VARCHAR2 (100)	Name of a site.
CDATA3	VARCHAR2 (100)	Not used.
CDATA4	VARCHAR2(6)	Not used.
CDATA5	VARCHAR2(6)	Not used.
CDATA6	VARCHAR2 (240)	Error messages.

Column name:	Data type:	Description:
NDATA1	NUMBER(8)	Error codes.

Index

Index name:	Unique?:	Columns indexed:
U_SITE_TABLE_TYPE	Yes	SITEID, TABLEINDX, SUBTYPE

REPGROUP

The CTSRP.REPGROUP table stores information about accounts that are in replication.

A row is created in this table when a site becomes a Replication Master site or Replication Subordinate site for an account.

Rows

One for each replicating account.

Columns

Column name:	Data type:	Description:
INDX	NUMBER(6)	Index assigned to the replicating account.
NAME	VARCHAR2(30)	Name of the account in replication.
WILLGIVE	CHAR(1)	Not used; always N.

Column name:	Data type:	Description:
OWNID	NUMBER(6)	DB_ID of the Replication Master site for the account.
PROMOTEID	NUMBER(6)	Reserved.
REPCHANGE	NUMBER(6)	Number of pending replication changes.
STATUS	CHAR(2)	Status of the account.

Indexes

Index name:	Unique?:	Columns indexed:
PK_GROUP_INDX	Yes	INDX
U_GROUP_NAME	Yes	NAME

REPGROUPOWN

The CTSRP.REPGROUPOWN table stores information about which site is the Replication Master site for an account. This table is only populated at the Multisite Master site.

A row is created in this table when an account is registered for replication.

Rows

One for each account in replication.

Columns

Column name:	Data type:	Description:
NAME	VARCHAR2(30)	Name of the account in replication.
OWNID	NUMBER(6)	DB_ID of the Replication Master site for the account.

Index

Index name:	Unique?:	Columns indexed:
PK_GROUPOWN_NAME	Yes	NAME

REPPARAMS

The CTSRP.REPPARAMS table is not used.

REPSITE

The CTSRP.REPSITE table stores information about sites particpating in replication. This table may have different contents at each site in the replication environment.

A row is created in this table when a site is designated as the Replication Master site of a protocol at the Replication Subordinate site, or when the Replication Subordinate site is invited into replication for an account by the Replication Master site.

Rows

One for each known replication site.

Columns

Column name:	Data type:	Description:
DBID	NUMBER(6)	Unique identifier of the site.
NAME	VARCHAR2(20)	User given name identifying the site.
STATUS	CHAR(2)	Status of the site.
DBNAME	VARCHAR2(40)	Global database name of the site.
SITETYPE	CHAR(1)	The type of the site: C — Multisite Master N — Non-Master R — Remote
ISHERE	CHAR(1)	Indicates if this is the local site: Y — Site is local. N — Site is not local.
SQLNET	VARCHAR2(60)	SQL* Net alias of the site.
PKEY	NUMBER(8)	CTSRP internal information.

Indexes

Index name:	Unique?:	Columns indexed:
PK_SITE_DBID	Yes	DBID
U_SITE_NAME	Yes	NAME

REPTABLE

The CTSRP.REPTABLE table stores information about tables in replication.

A row is created in this table when a table is added into replication for an account.

Rows

One for each table in replication.

Columns

Column name:	Data type:	Description:
INDX	NUMBER	Sequence of the replicating table.
NAME	NUMBER(7)	Name of the table in replication.
GROUPINDX	VARCHAR2(30)	Unique identifier of the replicating group, from CTSRP.REPGROUP.
STATUS	CHAR(2)	Replication status of the table.

Index

Index name:	Unique?:	Columns indexed:
PK_TABLE_INDX	Yes	INDX

10 CTRESOLVEREF Protocol

Overview

In order to use Resolve, the CTRESOLVEREF protocol must be imported. This protocol includes the following panels:

- CONTEXT – A placeholder panel that makes the installation of the VCT_ERRORSTATUS Type 4 panel possible.
- DISCREP_STATE
- DISCREP_TRANSITION
- INVESTIGATOR
- VCT_ERRORITEM
- VCT_ERRORSTATUS

Resolve uses the update tables, not the data tables, for panels in the CTRESOLVEREF protocol.

DISCREP_STATE panel

The DISCREP_STATE panel defines valid discrepancy statuses for the CTV_DSTATUS item in the VCT_ERRORSTATUS panel. For each record (discrepancy status), the items in this panel also determine whether:

- The discrepancy status is a terminal status.
- Users with specific access rights can modify discrepancy records in this status.
- Users with specific access rights can change records to this status.

The DISCREP_STATE panel is a Clintrial software Type 0 panel.

Rows

One per discrepancy status.

Columns

Column name:	Data type:	Description:
CODE	Text	Letter code representing a discrepancy status. Used by the DISCREP_TRANSITION panel and the Resolve program. Do not edit.
VALUE	Text	Name for display in dialog boxes and drop-down lists.
LABEL	Text	Not used for records supplied with Resolve. Lets you create a codelist for use in other Clintrial software modules. Accepts up to 40 characters.
LONGLABEL	Text	Not used for records supplied with Resolve. Lets you create a codelist for use in other Clintrial software modules. Accepts up to 80 characters.
END_FINAL	Number	Determines whether this is a terminal discrepancy status: 0 — no (not final) 1 — yes (final)
PROPOSE_CAN-_CHANGETO	Number	Indicates whether a user with the PROPOSE access right can set a discrepancy record to this discrepancy status: 0 — no (cannot change to) 1 — yes (can change to)
PROPOSE_CAN-_MODIFY	Number	Indicates whether a user with the PROPOSE access right can modify a discrepancy record that is in this discrepancy status: 0 — no (cannot modify) 1 — yes (can modify)
MANAGE_CAN-_CHANGETO	Number	Indicates whether a user with the MANAGE access right can set a discrepancy record to this discrepancy status: 0 — no (cannot change to) 1 — yes (can change to)

Column name:	Data type:	Description:
MANAGE_CAN-_MODIFY	Number	Indicates whether a user with the MANAGE access right can modify a discrepancy record that is in this discrepancy status: 0 — no (cannot modify) 1 — yes (can modify)
ALLOW_DUP	Number	Determines whether a new discrepancy record can be added if a duplicate is found in the same status when running validation. 0 — no (cannot add) 1 — yes (can add)

Indexes

No indexes are created for this panel other than the standard Clintrial software indexing.

Installed values

When the CTRESOLVEREF protocol is imported, the following values are supplied in the DISCREP_STATE panel:

Code:	Value:	End Final:	Propose Can Chg:	Propose Can Mod:	Manage Can Chg:	Manage Can Mod:	Allow Dup:
ACD	Autoclosed	1	0	0	0	1	1
ACM	Obsolete	1	0	0	0	0	1
CN	Confirmed As Is	1	0	0	1	0	0
LK	Linked	1	0	0	1	0	0
N	New	0	0	0	1	1	0
NO	No Action Needed	1	0	0	1	0	0

Code:	Value:	End Final:	Propose Can Chg:	Propose Can Mod:	Manage Can Chg:	Manage Can Mod:	Allow Dup:
RA	Resolution Applied	1	0	0	1	0	1
REI	Reissued	1	0	0	1	0	0
REL	Released	0	0	1	1	0	0
RI	Resolved Internally	1	0	0	1	0	0
RM	Resolved Manually	1	0	0	1	0	1
RP	Resolution Proposed	0	1	0	1	1	0
RTS	Ready To Send	0	1	0	0	1	0
S	Sent	0	0	0	0	1	0
SD	Source Deleted	1	0	0	1	0	0
UN	Unresolvable	1	0	0	1	0	0

DISCREP_TRANSITION panel

The CTRESOLVEREF.DISCREP_TRANSITION panel defines valid discrepancy transitions for the CTV_DSTATUS item in the VCT_ERRORSTATUS panel. This is a Clintrial software Type 0 panel.

Rows

One per permitted discrepancy transition.

Columns

Column name:	Data type:	Description:
START_STATE	Text	Code representing a discrepancy status (see DISCREP_STATE panel).
END_STATE	Text	Code of the same or another discrepancy status. Each DISCREP_TRANSITION record represents a possible transition from the START_STATE to the END_STATE. To change a given discrepancy status, your options are limited to the set of discrepancy statuses that has a relationship with it, as defined by a record in this panel.

Indexes

No indexes are created for this panel other than the standard Clintrial software indexing.

Installed values

When the CTRESOLVEREF protocol is imported, the following values are supplied in the DISCREP_TRANSITION panel:

Start State:	End State:	Start State:	End State:
ACD	RTS	RP	REI
CN	RP	RP	LK
LK	N	RP	RA
N	ACD	RP	UN
N	ACM	RP	CN
N	NO	RP	SD

Start State:	End State:	Start State:	End State:
N	LK	RP	RM
N	RI	RTS	ACD
N	REL	RTS	ACM
NO	N	S	ACD
REITO (See Note.)	N	S	ACM
REITO (See Note.)	REL	S	RP
REITO (See Note.)	RTS	S	REI
REL	ACD	S	LK
REL	ACM	S	CN
REL	RTS	S	RA
REL	RP	S	RM
RI	N	S	SD
RP	ACD	S	UN
RP	ACM	UN	RP

Note: Unlike the other codes in this table, REITO does not refer to a specific discrepancy status. Instead, this code indicates the discrepancy statuses that you can assign to the newly created discrepancy record when you reissue an existing discrepancy record. Only statuses with an REITO transition on file are available for the new discrepancy record when you change the status of an existing record to Reissued.

INVESTIGATOR panel

The INVESTIGATOR panel stores identifying data for investigators. One record is established in this panel for each investigator in the protocol.

The standard data discrepancy form installed with Resolve includes data from this panel. You can also include references to items in this panel if you redesign the standard data discrepancy form using PowerBuilder Version 7.0.2.

Rows

One per investigator.

Columns

Column name:	Data type:	Description:
INV_ID	Text	Stores the ID of an investigator.
INV_NAME	Text	Stores the name to print on data discrepancy forms next to the investigator ID.
INV_ADDRESS	Text	Not used by the standard data discrepancy form.
INV_TELEPHONE	Text	Not used by the standard data discrepancy form.
INV_FAX	Text	Not used by the standard data discrepancy form.
INV_CRA	Text	Not used by the standard data discrepancy form.
INV_PROTOCOL	Text	Stores the name of a specific protocol. If the investigator information in this record applies to all protocols, enter an asterisk (*).

Indexes

No indexes are created for this panel other than the standard Clintrial software indexing.

VCT_ERRORITEM panel

When you set up a protocol for Resolve, the VCT_ERRORITEM panel is copied from the CTRESOLVEREF protocol to that protocol. The VCT_ERRORITEM panel contains the names and values of items associated with an error. It also contains columns where proposed new values can be entered for review by data management. Each record in this panel identifies a data record (with the panel name and CT_RECID) and an item (by the item name).

The VCT_ERRORITEM panel is a Type 0 panel. The primary keys for this panel are:

- CTV_ERROR_ID
- CTV_PANEL
- CTV_DISCR_RECID
- CTV_ITEM_NAME

Rows

One per PL/SQL statement called for a VCT_ERRORITEM record.

Columns

Column name:	Data type:	Description:
CTV_ERROR_ID	Number	Unique identifier of the related VCT-_ERRORSTATUS record. Joins VCT-_ERRORITEM record(s) to a VCT-_ERRORSTATUS record.
CTV_PANEL	Text	Name of the panel containing data associated with this discrepancy record.
CTV_DISCR_RECID	Text	The CT_RECID item of the specific associated data record.
CTV_DISCREP-_PAGE_ID	Text	Value of the page context item in the associated data record.

Column name:	Data type:	Description:
CTV_DISCREP-_BLOCK	Text	Value of the block context item in the associated data record.
CTV_ITEM_NAME	Text	Name of the specific item that you want to associate with this discrepancy record.
CTV_ITEM_VALUE	Text	Value of the specific item at the time the discrepancy was detected.
CTV_NEW_VALUE	Text	Proposed new value for the specified item (or null).
CTV_NEW_VALUE-_GIVEN	Number	Check box indicating whether or not a new value has been provided. If not checked, indicates that no new value has been provided. Check to indicate that a New Value has been provided.
CTV_REPEAT_ID	Text	Optional description to provide contextual information about the item.
CTV_REASON	Text	Stores descriptive text or a code from the CTS_REASON_CODES codelist to describe the change made from the existing item value to the new value.

Index

In addition to standard Clintrial software indexing, the following index is created for this panel:

Index name:	Unique?:	Columns indexed:
VCT_ERRORITEM_UERR2IX	No	CTV_ERROR_ID

VCT_ERRORSTATUS panel

When you set up a protocol for Resolve, the VCT_ERRORSTATUS panel is copied from the CTRESOLVEREF protocol to that protocol. The VCT_ERRORSTATUS panel is a Type 4 panel.

Rows

One per discrepancy.

Columns

Column name:	Data type:	Description:
CTV_ERROR_ID	Number	Identifying number for a discrepancy record. ID numbers are unique within a protocol.
CTV_SOURCE	Text	The source of a discrepancy: • MANUAL • RULE • FLAG • CLASSIFY
CTV_CONTEXT1	Text	Value of first context item (subject identifier) from the data record that is the source of the discrepancy.
CTV_REMARKS	Text	Description of the data error; provided by the creator of the discrepancy record. For discrepancy records created by rules, the description provided within the rule is supplied.
CTV_REPLACE-_REMARKS	Text	Allows a reviewer to replace the description in the CTV_REMARKS column. When this column has a value, it displays on data discrepancy forms in place of CTV_REMARKS. For discrepancy records created by rules, the description provided within the rule is supplied.
CTV_ADDTL-_REMARKS	Text	Appended to CTV_REMARKS or to CTV_REPLACE_REMARKS on standard data discrepancy forms.
CTV_COMMENTS_1	Text	Internal comments from the first review (data management). Does not appear on standard data discrepancy forms.

Column name:	Data type:	Description:
CTV_COMMENTS_2	Text	Internal comments from a second review. Does not appear on standard data discrepancy forms.
CTV_DSTATUS	Text	Discrepancy status of the record (for example, New or Ready to Send). See the DISCREP_STATE panel.
CTV_STATUS- _CHANGE _DT	Date	Date and time that CTV_DSTATUS was last modified.
CTV_PRIORITY	Number	Stores a priority. For example, a three point scale (1, 2, 3) can indicate discrepancies of high, medium, or low priority. • For records created by rules, the priority can be assigned automatically. • For manually entered records, the person creating the record enters a priority.
CTV_CLOSED- _DATE	Date	Date and time the record is placed in a terminal discrepancy status (such as Resolution Applied or Linked).
CTV_CONFIRMA- TION_FLA	Number	Indicates that a resolution for the discrepancy record has been entered in the database, and only confirmation from the investigator is needed. In the More Detail window set the Query for Confirmation check box, or by the **Discrepancy** menu's **Query for Confirmation** command. Check this check box to indicate that necessary changes have already been made to the database. If this check box is left unchecked (the default), it indicates that no changes have been made to data records and that a resolution is needed from the investigator.
CTV_PROPOSAL- _TEXT	Text	Description of a proposed resolution. Useful for complex resolutions or if no VCT_ERRORITEM records are associated with the discrepancy record.

Column name:	Data type:	Description:
CTV_RESOLUTION-_CODE	Text	Stores descriptive text or a code from the CTS_REASON_CODES codelist to indicate a resolution reason (for example, original value confirmed or corrected value supplied). Supplied automatically from system parameters as follows: • From CTV_APPL_RES_CODE during applied proposed value processing. • From CTV_AUTOCLOSED_RES when validation autocloses a discrepancy record. • From CTV_OBSOLETE_RES when a discrepancy record's status is updated to Obsolete.
CTV_RESOLUTION-_COMM	Text	Message explaining an unusual resolution or the reason for the change.
CTV_LINKED-_ERROR_ID	Number	CTV_ERROR_ID of another discrepancy record. Used for records in Linked or Reissued discrepancy status to point to the discrepancy record that remains open.
CTV_RULE_NAME	Text	For records created by rules, stores the name of the rule.
CTV_FORM-_BATCH_NUM	Number	Identifier assigned to all records in a batch of data discrepancy forms. All records in a batch have the same batch number. This number can be used to recall the batch for reprinting, or to browse for and display records once the forms are returned.
CTV_FORM-_BATCH_DATE	Date	Date and time the batch was printed. All records in a batch have the same Form Batch Date.
CTV_BATCH-_ORDER_NUM	Number	A number that sorts discrepancy records in a batch into the order in which they appeared on printed data discrepancy forms.
CTV_PANEL	Text	Name of the panel containing the primary clinical data record associated with the discrepancy record.

Column name:	Data type:	Description:
CTV_DISCR_RECID	Text	CT_RECID of the primary data record associated with the discrepancy record.
CTV_ORCTABLE	Text	Oracle table where the primary data record was at the time the discrepancy was detected (that is, update or data).
CTV_DISCREP-_PAGE_ID	Text	Value of the page context item of the primary record.
CTV_DISCREP-_BLOCK	Text	Value of the block context item of the primary record.
CTV_REC_MOD-DATE	Date	Date of last modification of the primary record at the time the discrepancy was detected.
CTV_ERRDT	Date	Date and time the record was first inserted into the VCT_ERRORSTATUS panel.
CTV_ERRTYPE	Text	Process that created the discrepancy record: VALIDATE, FLAG, MERGE, MANUAL, or CLASSIFY.
CTV_ERRACT	Text	For records created by rules, error action (Report or Reject) taken by the rule.

Indexes

In addition to standard Clintrial software indexing, the following indexes are created for this panel:

Index name:	Unique?:	Columns indexed:
VCT_ERRORSTATUS_UERRIX	Yes	CTV_ERROR_ID
VCT_ERRORSTATUS-_UPANRULREC	No	CTV_PANEL, CTV_RULE_NAME, CTV_DISCR_RECID

11 CTL_REFERENCE Protocol

Overview

The CTL_REFERENCE protocol is a thesaurus-type protocol designed to help Lab Loader users calculate and maintain lab normal ranges and perform SI unit conversions using PL/SQL functions. This protocol includes the following panels:

- CTL_NORMAL_RANGE
- CTL_UNIT_CONVERSION

CTL_NORMAL_RANGE panel

The CTL_NORMAL_RANGE panel is a Clintrial software Type 0 panel that stores lab normal values and unit specifications for a particular lab, lab test, gender, weight range, patient age range, and effective date range.

Rows

One per lab normal range.

Columns

Column name:	Data type:	Description:
CTL$LAB_ID	VARCHAR2(40)	Lab Identifier.
CTL$TEST	VARCHAR2(40)	Test code.
CTL$START-_DATE	DATE	Effective start date of normal range test.
CTL$HIGH	NUMBER(18,4)	Test's high normal value.
CTL$LOW	NUMBER(18,4)	Test's low normal value.
CTL$UNIT	VARCHAR2(40)	Test units.

Column name:	Data type:	Description:
CTL$SEX	VARCHAR2(40)	Sex. You must enter a '.' if the normal ranges do not depend on sex.
CTL$AGE_LOW	NUMBER(7,2)	Age range minimum value. The value must be > 0.
CTL$AGE_HIGH	NUMBER(7,2)	Age range maximum value.
CTL$AGE_UNITS	VARCHAR2(40)	Age units.
CTL$WT_LOW	NUMBER(7,2)	Weight range minimum value. The value must be > 0.
CTL$WT_HIGH	NUMBER(7,2)	Weight range maximum value.
CTL$WT_UNIT	VARCHAR2(40)	Weight units.

Indexes

None.

CTL_UNIT_CONVERSION panel

The CTL_UNIT_CONVERSION panel is a Clintrial software Type 0 panel that stores unit conversions for a particular test.

Rows

One per unit conversion.

Columns

Column name:	Data type:	Description:
CTL$SRC_UNIT	VARCHAR2(40)	Unit to be converted FROM.
CTL$DEST_UNIT	VARCHAR2(40)	Unit to be converted TO.
CTL$TEST	VARCHAR2(40)	Test code. You must enter a '.' if the conversion does not depend on a Test code.
CTL$MULTIPLY	NUMBER(18,9)	Multiplication factor. Default value is 1.
CTL$ADD	NUMBER(18,9)	Additive constant. Default value is 0.

Indexes

None.

12 CT_MEDDRA Protocol

Overview

The CT_MEDDRA protocol is a thesaurus-type protocol specifically designed to hold MedDRA thesaurus data that can be used for coding. This protocol includes the following panels:

- L_PREF_TERM
- L_LOW_LEVEL_TERM
- L_MD_HIERARCHY
- L_SOC_TERM
- L_HLT_PREF_COMP
- L_HLGT_HLT_COMP
- L_HLGT_PREF_TERM
- L_HLT_PREF_TERM
- L_SOC_HLGT_COMP
- L_SOC_INTL_ORDER
- L_SPEC_CAT
- L_SPEC_PREF_COMP
- LLT_PT_SOC
- SYNONYMS
- STOPWORDS

In addition, six thesaurus views and three thesaurus coding algorithms are provided for use with the CT_MEDDRA protocol. A set of MedDRA translation functions are also provided. For information on the MedDRA functions, see "MedDRA functions" on page 352. For more information on MedDRA see the *Manage* section of *Manage, Classify, and Lab Loader.*

L_PREF_TERM panel

The L_PREF_TERM panel stores information about the Preferred Term.

Rows

One per record.

Columns

Column name:	Data type:	Description:
PT_ENGLISH_TEXT	VARCHAR2(100)	Normalized content of PT_NAME.
NULL_FIELD	VARCHAR2(1)	This field is null.
PT_CODE	NUMBER(8)	8-digit code to identify the Preferred Term.
PT_CODE_CHR	VARCHAR2(8)	Derived from PT_CODE for Character Text.
PT_COSTART-_SYM	VARCHAR2(21)	Symbol allocated by the COSTART© terminology.
PT_HARTS_CODE	NUMBER(8)	Code allocated by the HARTS© terminology.
PT_ICD10_CODE	VARCHAR2(8)	Code allocated by the 10th Revision of International Classification of Diseases, ICD-10©.
PT_ICD9_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, ICD-9.
PT_ICD9CM-_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, Clinical Modification, ICD-9-CM©.
PT_JART_CODE	VARCHAR2(6)	Code allocated by the J-ART terminology.

Column name:	Data type:	Description:
PT_NAME	VARCHAR2(100)	Full name of the Preferred Term.
PT_SOC_CODE	NUMBER(8)	The primary System Organ Class to which the Preferred Term is linked.
PT_WHOART-_CODE	VARCHAR2(7)	Code allocated by the WHO-ART© terminology.

Indexes

Index name:	Unique?:	Columns indexed:
PREF_TERM_DBIDX	Yes	PT_CODE
PT_VIEW_ENGLISH_THIDX	No	PT_ENGLISH_TEXT
IX1_PT02	Yes	PT_NAME
IX1_PT03	No	PT_SOC_CODE

L_LOW_LEVEL_TERM panel

The L_LOW_LEVEL_TERM panel stores information about the Low Level Term associated with the Preferred Term.

Rows

One per record.

Columns

Column name:	Data type:	Description:
LLT_ENGLISH_TEXT	VARCHAR2(100)	Normalized content of LLT_NAME.
LLT_CODE	NUMBER(8)	The 8-digit code to identify the Lowest Level Term.
LLT_NAME	VARCHAR2(100)	Full name of Low Level Term.
LLT_CODE_CHR	VARCHAR2(8)	Derived from LLT_CODE for Character Text.
LLT_COSTART-_SYM	VARCHAR2(21)	Symbol allocated by the COSTART© terminology.
LLT_CURRENCY	VARCHAR2(1)	Indicates whether the Low Level Term is current or noncurrent.
LLT_HARTS_CODE	NUMBER(8)	Code allocated by the HARTS© terminology.
LLT_ICD10_CODE	VARCHAR2(8)	Code allocated by the 10th Revision of International Classification of Diseases, ICD-10©.
LLT_ICD9_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, ICD-9.

Column name:	Data type:	Description:
LLT_ICD9CM-_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, Clinical Modification, ICD-9-CM©.
LLT_JART_CODE	VARCHAR2(6)	Code allocated by the J-ART terminology.
LLT_WHOART-_CODE	VARCHAR2(7)	Code allocated by the WHO-ART© terminology.
PT_CODE	NUMBER(8)	8-digit code to identify the Preferred Term.
PT_CODE_CHR	VARCHAR2(8)	Derived from PT_CODE for Character Text.

Indexes

Index name:	Unique?:	Columns indexed:
LOW_LEVEL_TERM_DBIDX	Yes	LLT_CODE
LLT_VIEW_ENGLISH_THIDX	No	LLT_ENGLISH_TEXT
IX1_PT_LLT02	No	LLT_NAME
IX1_PT_LLT03	No	PT_CODE

L_MD_HIERARCHY panel

The L_MD_HIERARCHY panel stores information from the MedDRA 1_MD_HIERARCHY table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLGT_CODE	NUMBER(8)	The 8-digit code to identify the High Level Group Term.
HLGT_NAME	VARCHAR2(100)	Full name of the High Level Group Term.
HLT_CODE	NUMBER(8)	8-digit code to identify the High Level Term.
HLT_NAME	VARCHAR2(100)	Full name of the High Level Term.
NULL_FIELD	VARCHAR2(1)	This field is null.
PRIMARY_SOC-_FG	VARCHAR2(1)	Flag set to Y/N to indicate Primary SOC.
PT_CODE	NUMBER(8)	8-digit code to identify the Preferred Term.
PT_NAME	VARCHAR2(100)	Full name of the Preferred Term.
PT_SOC_CODE	NUMBER(8)	The primary System Organ Class to which the Preferred Term is linked.
SOC_ABBREV	VARCHAR2(5)	System Organ Class abbreviation.
SOC_CODE	NUMBER(8)	8-digit code to identify the System Organ Class.

Column name:	Data type:	Description:
SOC_NAME	VARCHAR2(100)	Full name of the System Organ Class.

Indexes

IX1_MD_HIER_04

Index name:	Unique?:	Columns indexed:
IX1_MD_HIER_01	No	PT_CODE
IX1_MD_HIER_02	No	HLT_CODE
IX1_MD_HIER_03	No	HLGT_CODE
IX1_MD_HIER_04	No	SOC_CODE
IX1_MD_HIER_05	No	PT_SOC_CODE

L_SOC_TERM panel

The L_SOC_TERM panel stores information about the System Organ Classes.

Rows

One per record.

Columns

Column name:	Data type:	Description:
SOC_ENGLISH_TEXT	VARCHAR2(100)	Normalized content of SOC_NAME.
SOC_CODE	NUMBER(8)	The 8-digit code to identify the Lowest Level Term.
SOC_NAME	VARCHAR2(100)	Full name of the System Organ Class.
SOC_ABBREV	VARCHAR2(5)	System Organ Class abbreviation.
SOC_CODE_CHR	VARCHAR2(8)	Derived from SOC_CODE for Character text.
SOC_COSTART-_SYM	VARCHAR2(21)	Symbol allocated by the COSTART© terminology.
SOC_HARTS_CODE	NUMBER(8)	Code allocated by the HARTS© terminology.
SOC_ICD10_CODE	VARCHAR2(8)	Code allocated by the 10th Revision of International Classification of Diseases, ICD-10©.
SOC_ICD9_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, ICD-9.

Column name:	Data type:	Description:
SOC_ICD9CM-_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, Clinical Modification, ICD-9-CM©.
SOC_JART_CODE	VARCHAR2(6)	Code allocated by the J-ART terminology.
SOC_WHOART-_CODE	VARCHAR2(7)	Code allocated by the WHO-ART© terminology.

Indexes

IX1_MD_HIER_04

Index name:	Unique?:	Columns indexed:
SOC_TERM_DBIDX	Yes	SOC_CODE
IX1_SOC02	No	SOC_NAME

L_HLT_PREF_COMP panel

The L_HLT_PREF_COMP panel stores information from the MedDRA 1_HLT_PREF_COMP table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLT_CODE	NUMBER(8)	8-digit code to identify the High Level Term.
PT_CODE	NUMBER(8)	8-digit code to identify the Preferred Term.

Indexes

Index name:	Unique?:	Columns indexed:
IX1_HLT_PT01	No	HLT_CODE
IX1_HLT_PT02	No	PT_CODE

L_HLGT_HLT_COMP panel

The L_HLGT_HLT_COMP panel stores information from the MedDRA 1_HLGT_HLT_COMP table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLGT_CODE	NUMBER(8)	8-digit code to identify the High Level Group Term.
HLT_CODE	NUMBER(8)	8 digit code to identify the High Level Term.

Indexes

Index name:	Unique?:	Columns indexed:
IX1_HLGT_COMP_DBIDX	Yes	HLGT_CODE HLT_CODE
IX1_HLGT_HLT02	No	HLGT_CODE

L_HLGT_PREF_TERM panel

The L_HLGT_PREF_TERM panel stores information from the High Level Group Terms.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLGT_CODE	NUMBER(8)	8-digit code to identify the High Level Group Term.
HLGT_NAME	VARCHAR2(100)	Full name of the High Level Group Term.
HLGT_COSTART_SYM	VARCHAR2(21)	Symbol allocated by the COSTART© terminology.
HLGT_HARTS_CODE	NUMBER(8)	Code allocated by the HARTS© terminology.
HLGT_ICD10_CODE	VARCHAR2(8)	Code allocated by the 10th Revision of International Classification of Diseases, ICD-10©.
HLGT_ICD9_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, ICD-9.
HLGT_ICD9CM-_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, Clinical Modification, ICD-9-CM©.
HLGT_JART_CODE	VARCHAR2(6)	Code allocated by the J-ART terminology.

Column name:	Data type:	Description:
HLGT_WHOART-_CODE	VARCHAR2(7)	Code allocated by the WHO-ART© terminology.

Indexes

Index name:	Unique?:	Columns indexed:
IX1_HLGT01	No	HLGT_CODE
IX1_HLGT02	No	HLGT_NAME

L_HLT_PREF_TERM panel

The L_HLT_PREF_TERM panel stores information from the High Level Terms.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLT_CODE	NUMBER(8)	8-digit code to identify the High Level Term.
HLT_NAME	VARCHAR2(100)	Full name of the High Level Term.
HLT_COSTART_SYM	VARCHAR2(21)	Symbol allocated by the COSTART© terminology.
HLT_HARTS_CODE	NUMBER(8)	Code allocated by the HARTS© terminology.

Column name:	Data type:	Description:
HLT_ICD10_CODE	VARCHAR2(8)	Code allocated by the 10th Revision of International Classification of Diseases, ICD-10©.
HLT_ICD9_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, ICD-9.
HLT_ICD9CM-_CODE	VARCHAR2(8)	Code allocated by the 9th Revision of International Classification of Diseases, Clinical Modification, ICD-9-CM©.
HLT_JART_CODE	VARCHAR2(6)	Code allocated by the J-ART terminology.
HLT_WHOART-_CODE	VARCHAR2(7)	Code allocated by the WHO-ART© terminology.

Indexes

Index name:	Unique?:	Columns indexed:
IX1_HLT01	No	HLT_CODE
IX1_HLT02	No	HLT_NAME

L_SOC_HLGT_COMP panel

The L_SOC_HLGT_COMP panel stores information from the MedDRA 1_SOC_HLGT_COMP table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLGT_CODE	NUMBER(8)	8-digit code to identify the High Level Group Term.
SOC_CODE	NUMBER(8)	8-digit code to identify the System Organ Class.

Indexes

Index name:	Unique?:	Columns indexed:
SOC_HLGT_COMP_DBIDX	Yes	SOC_CODE HLGT_CODE
IX1_SOC_HLGT01	No	SOC_CODE
IX1_SOC_HLGT03	No	SOC_CODE HLGT_CODE **<<OK thru here!>>**

L_SOC_INTL_ORDER panel

The L_SOC_INTL_ORDER panel stores information from the MedDRA 1_SOC_INTL_ORDER table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
INTL_ORD_CODE	NUMBER(8)	Serial code for international System Organ Class sort order.
SOC_CODE	NUMBER(8)	8-digit code to identify the System Organ Class.

Indexes

Index name:	Unique?:	Columns indexed:
IX1_INTL_ORD01	Yes	INTL_ORD_CODE, SOC_CODE

L_SPEC_CAT panel

The L_SPEC_CAT panel stores information about the Special Search Category from the MedDRA 1_SPEC_CAT table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
SPEC_ABBREV	VARCHAR2(10)	Special Search Category abbreviation.
SPEC_CODE	NUMBER(8)	Serial code for Special Search Category.
SPEC_NAME	VARCHAR2(100)	Full name of Special Search Category.

Indexes

Index name:	Unique?:	Columns indexed:
SPEC_CAT_DBIDX	Yes	SPEC_CODE
IX1_SPEC02	No	SPEC_NAME
IX1_SPEC03	No	SPEC_ABBREV

L_SPEC_PREF_COMP panel

The L_SPEC_PREF_COMP panel stores information from the MedDRA 1_SPEC_PREF_COMP table.

Rows

One per record.

Columns

Column name:	Data type:	Description:
PT_CODE	NUMBER(8))	8-digit code to identify the Preferred Term..
SPEC_CODE	NUMBER(8)	Serial code for Special Search Category.

Indexes

Index name:	Unique?:	Columns indexed:
SPEC_PREF_COMP_DBIDX	Yes	SPEC_CODE PT_CODE
IX1_SPEC_PT02	No	PT_CODE SPEC_CODE

LLT_PT_SOC panel

The LLT_PT_SOC panel stores information about the aggregated data for Low Level Term (LLT), Preferred Term (PT) and System Organ Class (SOC).

Rows

One per record.

Columns

Column name:	Data type:	Description:
HLGT_CODE	NUMBER(8)	8-digit code to identify the High Level Group Term.
HLGT_NAME	VARCHAR2(100)	Full name of the High Level Group Term.
HLT_CODE	NUMBER(8)	8-digit code to identify the High Level Term.
HLT_NAME	VARCHAR2(100)	Full name of the High Level Term.
LLT_CODE	NUMBER(8)	8-digit code to identify the Low Level Term.
LLT_CODE_CHR	VARCHAR2(8)	Derived from LLT_CODE for Character Text.
LLT_ENGLISH_TEXT	VARCHAR2(100)	English Text for the Low Level Term.
LLT_NAME	VARCHAR2(100)	Full name of the Low Level Term.
PT_CODE	NUMBER(8)	8-digit code to identify the Preferred Term.
PT_CODE_CHR	VARCHAR2(8)	Derived from PT_CODE for Character Text.
PT_NAME	VARCHAR2(100)	Full name of the Preferred Term.

Column name:	Data type:	Description:
SOC_CODE	NUMBER(8)	8-digit code to identify the System Organ Class.
SOC_CODE_CHR	VARCHAR2(8)	Derived from SOC_CODE for Character Text.
SOC_NAME	VARCHAR2(100)	Full name of the System Organ Class.

SYNONYMS panel

The SYNONYMS panel stores information about thesaurus synonyms.

Rows

One per synonym.

Columns

Column name:	Data type:	Description:
CODE1	VARCHAR2(8)	Synonym code.
CODE2	VARCHAR2(8)	Synonym code.
ENGLISH_TEXT	VARCHAR2(100)	Full text of the synonym.

Index

Index name:	Unique?:	Columns indexed:
LLT_SYNONYMS_ENGLISH-_THIDX	No	ENGLISH_TEXT

STOPWORDS panel

The STOPWORDS panel stores stopwords that thesaurus algorithms use in coding.

Rows

One per stopword.

Column

Column name:	Data type:	Description:
ENGLISH_TEXT	VARCHAR2(100)	Full text of the stopword.

Index

Index name:	Unique?:	Columns indexed:
STOPWORDS_ENGLISH_THIDX	Yes	ENGLISH_TEXT

Thesaurus views

Six thesaurus views are provided for use with the CT_MEDDRA thesaurus protocol.

View name:	For coding on:	Based on panel:	Text item:	Code1 item:
PT_VIEW	Preferred Terms	L_PREF_TERM	PT_ENGLISH _TEXT	PT_CODE_C HR
PT_LLT_VIEW	Preferred Terms	L_LOW_LEVEL- _TERM	LLT_ENGLIS H_TEXT	PT_CODE_C HR
PT_SYNONYMS		SYNONYMS	ENGLISH_TE XT	CODE1
LLT_VIEW	Low Level Terms	L_LOW_LEVEL_ TERM	LLT_ENGLIS H-_TEXT	LLT_CODE_ CHR
LLT _SYNONYMS		SYNONYMS	ENGLISH- _TEXT	CODE1
STOPWORDS		STOPWORDS	ENGLISH- _TEXT	

Thesaurus algorithms

Three thesaurus coding algorithms are provided for use with the CT_MEDDRA thesaurus protocol.

Algorithm name:	For coding on:	Based on views:	Preferred view:
LLT_PT_SOC_ALG	LLT, PT, SOC	LLT_PT_SOC_VIEW	LLT_PT_SOC_VIEW
PT_ALG	Preferred Terms	PT_VIEW PT_LLT_VIEW PT_SYNONYMS STOPWORDS	PT_VIEW
LLT_ALG	Low Level Terms	LLT_VIEW LLT_SYNONYMS STOPWORDS	LLT_VIEW

13 PXFR_RECV Account

Note: This account has not been documented previously, because all the tables were identical to those in the CTS or CTSDD accounts.

IMPORT_PARAMS

The PXFR_RECV.IMPORT_PARAMS table stores information for each protocol in the process of being imported. This allows the protocol import process to be continued after reconciliation of codelists, flags and notes.

Rows

One for each protocol in the process of being imported.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol.
DIRECTORY	VARCHAR2(30)	Oracle directory for the dump file.
FILENAME	VARCHAR2(240)	Name of the dumpfile.
LOGDIR	VARCHAR2(30)	Oracle directory for the log file.
LOGNAME	VARCHAR2(240)	Name of the log file.
LOAD_DATA	NUMBER(1)	Include data in load? 1 — Yes 0 — No

Index

Index name:	Unique?:	Columns indexed:
IMPORT_PARAMS_PK	Yes	PROTOCOL

14 Lab Loader Tables

CTL_DUPLICATE table

The *protocol-name*.CTL_DUPLICATE table exists only in protocols that use Lab Loader to receive transferred data. This table stores record-level information about Lab Loader duplicates. It is created in a destination protocol account at the first attempt to transfer records into a table in the protocol.

Rows

One per duplicate record detected.

Columns

Column name:	Data type:	Description:
JOB_ID	NUMBER(10)	Job ID of the transfer job in which the duplicate was detected.
MAP_ID	NUMBER(15)	Name of the transfer map used for the transfer.
PROTOCOL	VARCHAR2(20)	Name of the source protocol from which the record was transferred.
PANEL	VARCHAR2(20)	Name of the destination panel where the duplicate resides.
CT_RECID_U	VARCHAR2(40)	CT_RECID of the duplicate record in the update table.
CT_RECID_D	VARCHAR2(40)	CT_RECID of the duplicate record in the data table. This column can be NULL.

Index

Index name:	Unique?:	Columns indexed:
CTL_DUPLICATE_INDX	No	JOB_ID

CTL_CONTROL_FILE table

The CTSDD.CTL_CONTROL_FILE table exists only if Lab Loader is installed on the server. This table stores information about control files in the Lab Loader control file library.

Rows

One per control file saved to the control file library.

Columns

Column name:	Data type:	Description:
PROTOCOL	VARCHAR2(20)	Name of the protocol with which the control file is associated.
NAME	VARCHAR2(20)	Name of the control file.
DESCRIP	VARCHAR2(240)	Text description of the control file. This column value can be NULL.
OBJECT_ID	NUMBER(15)	A unique identifier for the object, taken from the CTSDD.OBJINDX table.
MODDATE	DATE	Date and time at which the control file was created or last modified.
MODUSER	VARCHAR(20)	User account that created or last modified the control file.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
CTL_CONTROL_FILE_PK	Yes	PROTOCOL, NAME

CTL_MAP table

The CTSDD.CTL_MAP table exists only if Lab Loader is installed on the server. This table stores information about Lab Loader transfer maps.

Rows

One per transfer map.

Columns

Column name:	Data type:	Description:
MAP_ID	NUMBER(15)	A unique internal tracking number. Used to index CTSDD.CTL_MAP_ITEM.
NAME	VARCHAR2(20)	Name of the transfer map.
DEST_PROTOCOL	VARCHAR2(20)	Name of the destination protocol.
SRC_PROTOCOL	VARCHAR2(20)	Name of the source protocol.
SRC_PANEL	VARCHAR2(20)	Name of the source panel.
DESCRIP	VARCHAR2(240)	Text of the map comment field. This column value can be NULL.

Column name:	Data type:	Description:
STATUS	NUMBER(1)	Status of the transfer map. 0 — The map is known to be valid. 1 — The map is known to be invalid. For example, the source protocol, source panel, or destination protocol has been deleted after the map was created, or the Data types of items no longer match. 2 — The map is outdated. One of the source or destination protocols or panels has been edited more recently than the map. 3 — The map is incomplete. That is, the map was saved before all necessary changes were entered. For example, panel keys have not yet been mapped.
MODDATE	DATE	Date and time at which the transfer map was created or last modified.
MODUSER	VARCHAR(20)	User account that created or last modified the transfer map.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
CTL_MAP_PK	Yes	NAME, DEST_PROTOCOL, SRC_PROTOCOL, SRC_PANEL

CTL_MAP_ITEM table

The CTSDD.CTL_MAP_ITEM table exists only if Lab Loader is installed on the server. This table stores information about the items in Lab Loader transfer maps.

Rows

One for each item within each transfer map.

For each transfer map, there is one row for:

- Each non-context, non-system item with every destination panel.
- Each context item for the destination protocol, if any of the destination panels is Type 1-4. There is only one entry for each context item, even if that context item appears in more than one of the destination panels.
- The [Destination Protocol] transfer map item.

Columns

Column name:	Data type:	Description:
MAP_ID	NUMBER(15)	The MAP_ID for a transfer map, taken from the CTSDD.CTL_MAP table. This column has the same value for all entries for a given map.
DEST_PANEL	VARCHAR2(20)	Identifies the panel to which the source expression is being mapped. For [Destination Protocol] the value is "-".
DEST_ITEM	VARCHAR2(20)	Identifies the item to which the source expression is being mapped. For [Destination Protocol] the value is "[Dest Protocol]".

Column name:	Data type:	Description:
SRC_EXPR_TYPE	NUMBER(1)	With SRC_EXPR, determines the value that maps into a given destination item. It can have one of the following values: 0 — The destination item is unmapped. Its value will be NULL after the transfer has been formed. 1 — The value of the destination item is taken from an item within the source panel. 2 — The value of the destination item is the job ID of the transfer job.
SRC_EXPR	VARCHAR2(240)	With SRC_EXPR_TYPE, determines the value that maps into a given destination item. It can have one of the following values: NULL — The destination item is unmapped. *Source-item-name* — The value of the destination item is taken from this item. [Job Id] — The value of the destination item is the job ID of the transfer job.
DB_ID	NUMBER(5)	Obsolete.

Index

Index name:	Unique?:	Columns indexed:
CTL_MAP_ITEM_PK	Yes	MAP_ID, DEST_PANEL, DEST_ITEM

Part II: Programming

15 Using PL/SQL in the Clintrial Software

PL/SQL basics

What is PL/SQL?

PL/SQL is Oracle's procedural language extension to SQL. Working with the Clintrial software, you can use PL/SQL in:

- Derivations that are attached to panels.
- Rules that are attached to panels.
- Data-entry processing procedures that are attached to page templates, page sections, or items.

For complete information about PL/SQL programming, see your Oracle documentation.

What is a block?

The basic programming unit in PL/SQL is a *block*. A PL/SQL block is a unit of PL/SQL code that groups together logically related declarations and statements. A block consists of the following sections:

Section:	Contents:
Header	Mandatory section for named blocks. Includes the name, parameter list, and RETURN clauses (for a function only). Does not apply to anonymous blocks.
Declaration	Optional section that declares variables, cursors, and sub-blocks that are referenced in the execution and exception sections.
Execution	Mandatory section that contains IF_THEN_ELSEs, LOOPs, assignments, and calls to other blocks. Must contain at least one executable statement.
Exception	Optional section that handles exceptions to normal processing.

There are two types of blocks:

- *Anonymous block* – A block that does not have a name, and cannot be called by other PL/SQL code. An anonymous block can be run as a script or can be nested in a named block or in another anonymous block.
- *Named block* – A block that has a name, and can be called by other PL/SQL code. There are two types of named blocks: stored functions and stored procedures.

Stored functions

A stored *function* returns a single value. You can pass information to a function using the function's parameter list.

The format of a stored function is:

FUNCTION *function-name* *(parameter-list*)
RETURN *datatype*
IS
declaration-statements

BEGIN
executable-statements

EXCEPTION
exception-handler-statements

END *function-name*;

Example: PL/SQL function

The following example is a stored function that compares the value of two dates and determines whether the second date is greater than the first date:

```
FUNCTION date_compare
(date1 DATE,
date2 DATE)
return BOOLEAN IS

BEGIN
if (date2 > date1) then
return TRUE;
else
return FALSE;
end if;

END date_compare;
/
```

Stored procedures

A stored *procedure* performs one or more actions. You can pass information into and out of a procedure using its parameter list.

The format of a stored procedure is:

PROCEDURE *procedure-name*(*parameter-list*)
IS
declaration-statements

BEGIN
executable-statements

EXCEPTION
exception-handler-statements

END *procedure-name*;

Example: PL/SQL procedure

The following example is a stored procedure that sets the focus to the AECODE item:

```
CREATE OR REPLACE PROCEDURE item_focus
(i_protocol VARCHAR2,
i_panel VARCHAR2,
i_table VARCHAR2,
i_colname VARCHAR2,
i_ct_recid VARCHAR2,
i_colvalue VARCHAR2,
i_itemvalues VARCHAR2,
o_result OUT INTEGER,
o_new_value OUT VARCHAR2,
o_message OUT VARCHAR2,
o_new_itemvalues OUT VARCHAR2)
IS

BEGIN
ct_event.item_focus('AECODE');

o_result := 1;
o_message := 'Cursor moves to AE code.';

END item_focus;
/
```

What is a package?

A *package* is a saved collection of PL/SQL objects that are grouped together within a BEGIN-END syntax. The objects can be cursors, variables, constants, exception names, PL/SQL table and record TYPE statements, procedures, and functions.

A package consists of:

- A package specification
- A package body

Package specification

The *package specification* contains the definition of public elements in the package, that is, elements that can be referenced from outside the package. A package can contain:

- Variable declarations
- TYPE declarations
- Exception declarations
- Cursor specifications
- Function specifications
- Procedure specifications

If the package specification includes specifications for a cursor, function, or procedure, then there must be a package body.

The format of a package specification is:

PACKAGE *package-name*
IS
declarations-of-variables-and-types]

specifications-of-cursors

specifications-of-functions-and-procedures

END *package-name*;

Example: package specification

The following example is a package specification for a package named ct_valprocs:

```
CREATE OR REPLACE PACKAGE ct_valprocs
IS

FUNCTION date_compare
(date 1 DATE,
date 2 DATE)
return BOOLEAN;

FUNCTION item_not_null
(item1 VARCHAR2)
return BOOLEAN;

END ct_valprocs;
/
```

Package body

The *package body* contains the code that implements the package specification. There must be a package body if the package specification includes specification of cursors, functions, or procedures.

The format of a package body is:

PACKAGE BODY ***package_name***
IS
declarations-of-variables-and-types

specification-and-SELECT-statements-of-cursors

specification-and-body-of-functions-and-procedures

BEGIN
executable-statements

EXCEPTION
exception-handler-statements

END *package-name*;

Example: package body

The following example is a package named ct_valprocs, which stores two functions, named date_compare and item_not_null:

```
CREATE OR REPLACE PACKAGE BODY ct_valprocs
IS

FUNCTION date_compare (date1 DATE, date2 DATE)
return BOOLEAN
IS

BEGIN

if (date2 > date1) then
return TRUE;
else
return FALSE;
end if;

END date_compare;

FUNCTION item_not_null
(item1 VARCHAR2)
return BOOLEAN
IS

BEGIN
if (item1 is not null) then
return TRUE;
else
return FALSE;
end if;
END item_not_null;

END ct_valprocs;
/
```

PL/SQL in the Clintrial software

How the term "function" is used

In this chapter, and in the Clintrial software user interface, the term "function" is used to refer to a PL/SQL stored *function, procedure,* or *package*. However, where the distinction between these different types of objects is essential, the specific type of object is identified.

Types of functions

You can use the following types of functions:

- *Clintrial software functions,* which are delivered with the Clintrial software in packages. These packages are listed below and are described in detail in Chapter 16. There are six types of Clintrial software functions:
  - Basic functions
  - String functions
  - Resolve functions
  - Lab Loader functions
  - MedDRA functions
  - Event utility functions
- *Site-specific functions*, which are defined by your site for use by multiple protocols. For example, if you create a generic procedure to compare two dates, you would probably set it up as site-specific because it would be useful for multiple protocols. The next section describes how to create site-specific functions.
- *Protocol-specific functions*, which are defined by your site for use by specific protocols. For example, if a protocol is related to a drug used for joint pain, and a procedure determines a score index for joint pain and joint swelling, you would probably set up the procedure as protocol-specific. The next section describes how to create protocol-specific functions.

Copying functions between instances

If you are using Multisite, you can copy customized functions *between* database instances. One effect of the PROC_SITE_ACCOUNT and PROC_ACCOUNT parameters (described on page 298) is to determine which customized functions are available for copying in Multisite.

For more information about copying functions between instances, see *Multisite*.

Packages delivered with the Clintrial software

the Clintrial software is delivered with predefined PL/SQL packages containing variables, functions, and procedures that you can reference when using PL/SQL for Clintrial software derivations, rules, or data-entry processing procedures. These packages are listed in the following table:

PL/SQL package:	Contents:
CT_GLOBAL	Clintrial software variables, described on page 294. Available for derivations and rules (but not data-entry processing procedures).
CT_FUNC	Basic functions, described in "Basic functions" on page 310. Available for derivations, rules, and data-entry processing procedures.
CT_STRING	String functions, described in "String functions" on page 329. Available for derivations, rules, and data-entry processing procedures.
CT_PROC- _ACCOUNT	Privilege functions, described in "Privilege functions" on page 344. Available for granting privileges to site-specific or protocol-specific functions.
CTV_CORE	Resolve functions, described in "Resolve functions" on page 345. Available for derivations and rules (but not data entry processing procedures).
CTL_CORE	Lab Loader functions, described in "Lab Loader functions" on page 345. Available for derivations and rules (but not data-entry processing procedures).
CT_MEDDRA	MedDRA functions, described in "MedDRA functions" on page 352. Available for derivations and rules (but not data-entry processing procedures). Also used in Retrieve queries for clinical data coded with a MedDRA thesaurus.
CT_EVENT	Event utility functions, described in "Event utility functions" on page 359. Available for data-entry processing procedures (but not derivations and rules).

Calling a function

To call a function (or procedure) from within the Clintrial software, use the following format:

package-name.function-name

For the packages delivered with the Clintrial software (for example, CT_FUNC), public synonyms have already been created. Thus, you do not need to specify the owner account.

To call a site-specific or protocol-specific function (or procedure) from within the Clintrial software, use the following format:

owner-account.function-name

If the function (or procedure) is in a package, use the following format:

owner-account.package-name.function-name

Page 302 describes how to create public synonyms. If you create a public synonym for a function, procedure, or package, then you can use the public synonym from within the Clintrial software.

What are Clintrial software variables?

The CT_GLOBAL package delivered with the Clintrial software contains variables that you can reference in rules and derivations. For example, ct_global.cts$panel is a Clintrial software variable that points to the current panel; you can specify ct_global.cts$panel instead of the specific panel.

Note: The Clintrial software variables cannot be used in data-entry processing procedures.

How to use Clintrial software variables

To use a Clintrial software variable within a rule or derivation, you must preface the Clintrial software variable with ct_global. For example:

ct_global.*variable-name*

Process-related variables

The following process-related variables are available:

Clintrial software variable:	Description:
cts$protocol	Name of the current protocol.
cts$panel	Name of the current panel.
cts$table	Table type of the current Clintrial software table: • UPDATE for the *panel-name*_UPDATE table • DATA for the *panel-name*_DATA table
cts$tbl	Name of the current table, for example, MEDIKA_CLINICAL.ADV_UPDATE.
cts$err_date	Date on which an error occurred for the current record. This is the value of the ERRDT column in the *protocol-name*.ERRORLOG table.
cts$err_type	Process during which an error occurred for the current record; this value is always VALIDATE. This is the value of the ERRTYPE column in the *protocol-name*-.ERRORLOG table.

Record-related variables

The following record-related variables are available:

Clintrial software variable:	Description:
cts$ct_recid	Value of the CT_RECID (a system item) for the record.
cts$subject_id	Value of the SUBJECT_ID (a system item) for the record.
cts$subject	Value of the subject item, as a text string.
cts$block	Value of the block key item, as a text string.
cts$page	Value of the page key item, as a text string.

Clintrial software variable:	Description:
cts$block_repeat	Value of the block repeat key item, as a text string.
cts$page_repeat	Value of the page repeat key item, as a text string.
cts$rec_moddate	Value of the system item MERGE_DATETIME.

Rule-related variables

The following rule-related variables are available:

Clintrial software variable:	Description:
cts$rule_name	Name of the current rule.
cts$err_msg	Message text for the current rule.
cts$err_action	Error action (REPORT or REJECT) resulting from the current rule.

Derivation-related variable

The following derivation-related variable is available:

Clintrial software variable:	Description:
cts$deriv_name	Name of the current derivation.

"this" identifier

Within a Clintrial software derivation or rule, you can refer to any item in the current record by using the Clintrial software identifier "this" in the following format:

this.*item-name*

The "this" identifier is a convenience. If you do not specify "this", you must specify the whole table name, including the protocol account name.

For example, suppose that you want to check whether values exist for the TEMP and PULSE items in the update table or data table of the VITALS panel. You can write the following rule in the Clintrial software:

```
ct_func.msg_if_empty('TEMP, PULSE',
ct_string.make_list(this.temp,this.pulse));
```

This rule would apply regardless of whether validation is run for records in the update table or the data table.

You cannot use the "this" identifier in CONTEXT_INIT, PANEL_INIT, CONTEXT_END, and PANEL_END derivations, because there is no current record. For the CONTEXT_INIT and PANEL_INIT derivations, all values are NULL. For the CONTEXT_END and PANEL_END derivations, all values are the same as for the last record that was processed.

Note: The "this" identifier cannot be used in the text of PL/SQL functions that are stored outside of the Clintrial software. Thus, it cannot be used in data-entry processing procedures.

Converting values

The following functions perform conversion of values to an appropriate Clintrial software or Oracle format:

- CT_STRING.CHAR_TO_DATE
- CT_STRING.CHAR_TO_DATETIME
- CT_STRING.CHAR_TO_FLOAT
- CT_STRING.DATETIME_TO_CHAR
- CT_STRING.DATE_TO_CHAR
- CT_STRING.FLOAT_TO_CHAR

These functions are described in detail in Chapter 16.

Use these functions to pass values to and from the Clintrial software, for example, when writing data entry processing procedures. Additionally, if you use Multisite, you can use these functions to ensure that values are not interpreted according to Oracle settings (such as the decimal separator or the NLS_DATE_FORMAT parameter), which may vary on different servers.

Site-specific and protocol-specific functions

Where to store customized functions

The Clintrial software does not provide support for customizations of the content of the CTPROC account (or any other system account). The CTPROC account is a privileged Oracle account containing Clintrial software code. To store customized functions, you should set up special Oracle accounts.

You can set up the following Oracle accounts for customized functions:

- An Oracle account to store site-specific, systemwide functions that are available to all protocols in the Clintrial software database instance.
- For each protocol, an Oracle account to store protocol-specific functions that are available to that protocol only. You may also choose to store protocol-specific functions for more than one protocol in the same Oracle account.

Note: You can name these accounts whatever you want. This guide uses as examples an account named CTSITEPROC to store site-specific functions, and accounts named KA001PROC and FR001PROC to store protocol-specific functions.

Required system privileges

An Oracle account that contains site-specific or protocol-specific functions should have the following Oracle system privileges:

- CREATE SESSION
- CREATE PROCEDURE
- CREATE PUBLIC SYNONYM
- DROP PUBLIC SYNONYM
- ALTER ANY PROCEDURE

Parameter settings

If you set up an Oracle account to store site-specific functions, the protocol parameter PROC_SITE_ACCOUNT should specify the name of that account. If you set up an Oracle account to store protocol-specific functions, the protocol parameter PROC_ACCOUNT should specify the name of that account.

The accounts specified by these parameters are automatically given the following Oracle object privileges:

- When a protocol is created, the SELECT privilege on the protocol's SUBJECT_BLOCK, SUBJECT_PAGE, TAGS, and TAGS_AUDIT tables
- When a panel is installed, the SELECT privilege on update table, data table, and audit table for the panel, and on the *panel-name*_ALL view

Note: If you change the setting of PROC_SITE_ACCOUNT or PROC_ACCOUNT, then the Clintrial software asks if you want to retroactively grant privileges. If you answer Yes, then the preceding SELECT privileges are granted to the protocol (or all protocols).

Additionally, the settings of the PROC_SITE_ACCOUNT and PROC_ACCOUNT parameters:

- Affect the GRANT_EXECUTE_PRIVS function (described on page 301).
- Determine which customized functions are listed in Design as available when you create Clintrial software derivations and rules.
- Determine which customized functions are available for copying in Multisite.

Note: It is not essential that you set the PROC_SITE_ACCOUNT or PROC_ACCOUNT parameters to use customized functions; however, it is recommended.

Packaging customized functions

You can store customized functions as stored functions or stored procedures, or you can store them in packages. See your Oracle documentation for information about the advantages of using PL/SQL packages.

Packages allow you to group together particular types of functions and procedures. For example, you could create the following packages:

- A package to store site-specific functions and procedures used by Clintrial software validation procedures (derivations and rules)
- A package to store site-specific Clintrial software data-entry processing procedures
- For each protocol, a package to store protocol-specific functions and procedures used in Clintrial software validation procedures (derivations and rules)
- For each protocol, a package to store protocol-specific Clintrial software data-entry processing procedures

The following table is an example of accounts and packages that store site-specific functions and protocol-specific functions for the KA001 protocol and FR001 protocols:

Oracle account:	Parameter setting:	Packages:
CTSITEPROC	PROC_SITE_ACCOUNT = CTSITEPROC	ct_valprocs ct_deprocs
KA001PROC	PROC_ACCOUNT = KA001PROC	ka001_valprocs ka001_deprocs
FR001PROC	PROC_ACCOUNT = FR001PROC	fr001_valprocs fr001_deprocs

Steps to create a function

To create a site-specific or protocol-specific function for use with the Clintrial software, you can use any third-party tools that enable you to write, compile, and work with PL/SQL. You must do the following:

1. Write and save the function as a script file.
2. Compile the script file.
3. Grant the EXECUTE privilege on the function.

   ***Note**:* This is necessary for functions called by derivations or rules, but not for functions called by data-entry processing procedures.
4. Optionally create a public synonym for the function.

Compiling a function

To compile a site-specific or protocol-specific function, log in to the account that should own the package, and compile the file(s) containing the function.

Note: The account in which you compile a function is referred to as the *owner-account* in this guide.

For example, suppose that you write a site-specific function to be owned by the CTSITEPROC account (or whatever you name the Oracle account that you created to store site-specific functions). You create and save the following two files and store them in your \Ct43 folder:

- ct_valprocs.sqh

- ct_valprocs.sql

To compile the function:

1. Log in to the CTSITEPROC account.
2. Compile the function, for example:

   SQL> **@ C:\ct43\ct_valprocs.sqh;**
   SQL> **@ C:\ct43\ct_valprocs.sql;**

For a package containing protocol-specific functions, you would log in to the account that stores the protocol-specific functions and perform the compilation.

Granting the EXECUTE privilege

Each protocol whose derivations or rules refer to a site-specific or protocol-specific function must have the EXECUTE privilege on the function. To grant the EXECUTE privilege, you can use the GRANT_EXECUTE_PRIVS in the CT_PROC_ACCOUNT package that is delivered with the Clintrial software.

Note: You do not need to grant the EXECUTE privilege to data-entry processing procedures; these procedures are called by Clintrial software code that automatically has the required privileges. Also, the packages delivered with the Clintrial software (for example, CT_FUNC) have already been granted the EXECUTE privilege to PUBLIC.

To grant the EXECUTE privilege for a site-specific function:

1. Log in to the account that stores site-specific functions.
2. Enter the following:

   SQL> **execute ct_proc_account.grant_execute_privs ('*function-name*');**

   The Clintrial software grants the EXECUTE privilege on the function to all protocols whose system parameter PROC_SITE_ACCOUNT is set to the current account.

To grant the EXECUTE privilege for a protocol-specific function:

1. Log in to the account that stores protocol-specific functions.
2. Enter the following:

   SQL> **execute ct_proc_account.grant_execute_privs ('*function-name*');**

   The Clintrial software grants the EXECUTE privilege to all protocols whose protocol parameter PROC_ACCOUNT is set to the current account.

For customized functions stored in accounts that are not specified by the PROC_SITE_ACCOUNT or PROC_ACCOUNT, you must explicitly grant the EXECUTE privileges to PUBLIC or to specific protocols.

Creating public synonyms

You can create public synonyms for the following:

- Stored functions and stored procedures that are not in a package
- Packages (You can create a synonym for a package itself, but you cannot create synonyms for the functions and procedures that are in the package.)

Although it is not mandatory, it is recommended that you create a public synonym for each site-specific or protocol-specific package. If you do so, then references to the package from Clintrial software derivations, rules, and data-entry processing procedures, do not need to include the owner account.

Note: For the packages delivered with the Clintrial software (for example, CT_FUNC), public synonyms have already been created.

To create a public synonym:

1. Log in to the Oracle account created to store site-specific or protocol-specific functions. (These accounts should have the privileges needed to create and drop public synonyms.)
2. Enter the following commands:

   SQL> **drop public synonym *synonym-name*;**
   SQL> **create public synonym *synonym-name* for**
   SQL> ***owner-account.package-name*;**

   ***Note*:** If the public synonym does not already exist, you do not need to drop it before creating it.

For example, the following statement creates the public synonym ct_valprocs for the ct_valprocs package owned by (compiled by) the ctsiteproc account:

SQL> **create public synonym ct_valprocs for ctsiteproc.ct_valprocs;**

Suppose that the ct_valprocs package includes a date_compare function. Once you have created the public synonym ct_valprocs, then from within the Clintrial software, you can write a rule as follows:

```
ct_valprocs.date_compare(this.stopdate,this.startdate);
```

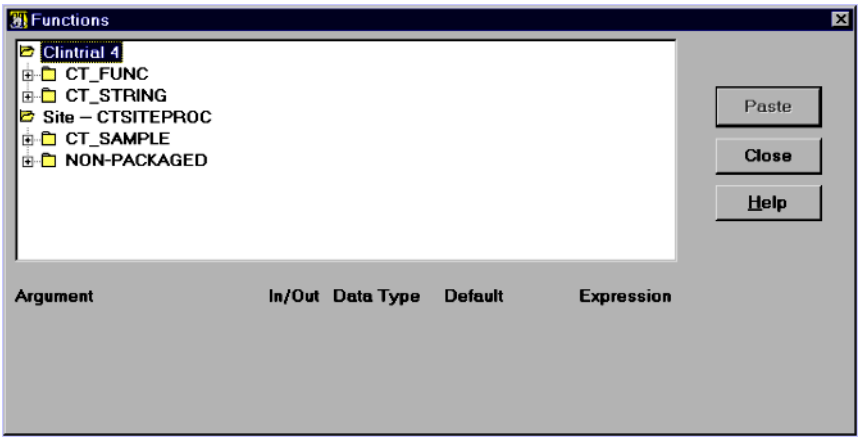
If you had not created a public synonym, you would need to include the owner account when calling the package:

```
ctsiteproc.ct_valprocs.date_compare(this.stopdate, this.startdate);
```

PL/SQL in derivations and rules

Using functions in derivations and rules

When you are specifying the text of a derivation or rule in Design, you can use the **Syntax** menu's **Functions** command to cut and paste references to Clintrial software functions, site-specific functions, and protocol-specific functions. The Functions dialog box opens:



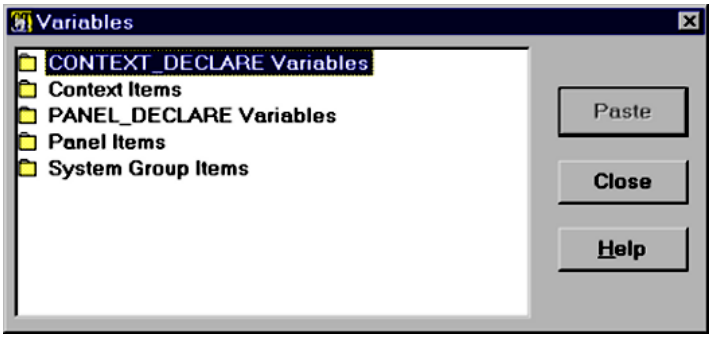
The following table describes the entries in the Functions dialog box:

Entry:	What is available:
Clintrial 4	Clintrial software functions contained in the following packages: • CT_FUNC • CT_STRING ***Note:*** The CT_EVENT package is not in this list because it is only available for data-entry processing procedures, and not for derivations or rules.

Entry:	What is available:
Site - *account-name*	Site-specific functions that are owned by the account identified by the system parameter PROC_SITE_ACCOUNT. Each package owned by the account is listed. For each package, each function or procedure in that package is listed. The NONPACKAGED entry lists stored functions and procedures that are owned by the account, but are not stored in packages.
Protocol - *account-name*	Protocol-specific functions that are owned by the account identified by the protocol parameter PROC_ACCOUNT. Each package owned by the account is listed. For each package, each function or procedure in that package is listed. The NONPACKAGED entry lists stored functions and procedures that are owned by the account, but are not stored in packages.

Using variables in derivations and rules

When you are specifying the text of a derivation or rule in Design, you can use the **Syntax** menu's **Variables** command to cut and paste references to items. The Variables dialog box opens:



The following table describes the entries in the Variables dialog box:

Entry:	What is available:
CONTEXT_DECLARE Variables	Name of each temporary variable that is declared by the CONTEXT_DECLARE derivation.
Context Items	Name (prefaced by "this") of each item in the CONTEXT panel.
PANEL_DECLARE Variables	Name of each temporary variable that is declared by the PANEL_DECLARE derivation.
Panel Items	Name (prefaced by "this") of each item in the current panel.
System Group Items	Name (prefaced by "this") of each system item.

Validation procedures

When you install a Clintrial software panel (or implement revisions to it), the Clintrial software automatically builds a validation procedure for the panel. The validation procedure is named VLD*_panel-name*. For more information about validation procedures, see the *Design* section of *Admin and Design*.

You should compile functions that are called by validation procedures *before* the validation procedures are compiled. Otherwise, the following situation occurs:

1. The validation procedure is compiled and has the status invalid, because the functions that the validation procedure calls have not been compiled.
2. You compile the functions.
3. You cannot run validation because the validation procedure status is invalid.
4. You must recompile the validation procedure using the **Panel** menu's **Compile** command, or the SQL statement ALTER PROCEDURE.

   To use the ALTER PROCEDURE statement, enter the following:

   SQL> **alter procedure *owner-account.*VLD*_panel-name* compile;**

If you import a protocol whose derivations and rules are valid, the derivations and rules remain valid in the new protocol; however, the validation procedure may be invalid. For example, if a function called by a derivation or rule is not compiled, then the validation procedure is invalid. You must compile the

function, grant the EXECUTE privilege on it, and recompile the validation procedure using the **Panel** menu's **Compile** command (or the SQL statement ALTER PROCEDURE).

16 Using Clintrial Software Functions

Basic functions

What is a basic function?

A *basic function* is a Clintrial software function that is defined in the CT_FUNC package, which is delivered with the Clintrial software. The basic functions are available for use in Clintrial software derivations, rules, and data-entry processing procedures.

Note: The Design interface for creating derivations and rules may list procedures that are not described in this guide. Procedures that are not described in this guide are intended for Clintrial software internal use only.

How to call a basic function

To call a basic function, preface the function name with CT_FUNC (the package name). For example:

```
ct_func.block_has_data(ct_global.cts$protocol,
ct_global.cts$block,null,ct_global.cts$subject_id)
```

BLOCK_HAS_DATA

Function

FUNCTION BLOCK_HAS_DATA(
i_protocol IN VARCHAR2,
i_block_key IN VARCHAR2,
i_block_repeat_key IN VARCHAR2,
i_subject_id IN INTEGER)

RETURN OUT INTEGER

Action

Determine if a specified block has existing records.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	Name of the protocol.
i_block_key	Block key value.
i_block_repeat-_key	Block repeat key value.
i_subject_id	Value of the system item SUBJECT_ID. For a derivation or rule, use the variable ct_global.cts$subject_id. If i_subject_id is Null, the function checks for any subjects.

Returned values

If the block does not have existing records, returns 0.

If the block has existing records, returns 1.

If the protocol does not exist, returns -1.

Example

The following example determines if a block has data:

```
ret_status := ct_func.block_has_data(ct_global.cts$protocol, ct_global.cts$block,
null,ct_global.cts$subject_id);
```

CLOSE_LOOKUP

Function

FUNCTION CLOSE_LOOKUP(
i_lookup_id IN NUMBER)

Action

Close the cursors that were opened by a lookup.

Argument

This function uses the following argument:

Argument:	Description:
i_lookup_id	Lookup number returned by the CT_FUNC.LOOKUP_VARS function.

Example

The following example uses the CLOSE_LOOKUP function:

```
// the user has looked up value and wants to close the cursor

io_lookuip_id integer;
o_value_list varchar2(240);
tbl_found varchar2(20);

tbl_found := ct_func.lookup_vars(ct_global.cts$protocol,'LABS',
  'UD',('TEST_NAME = WBC'),'SUBJECT =, VISNO = ', '120,2',
   'TEST_NAME, UNITS, RESULT' o_value_list, io_lookup_id);

ct_func.close_lookup(io_lookup_id);
```

CONVERT_DATE

Function

FUNCTION CONVERT_DATE(
i_year_part IN VARCHAR2
i_month_part IN VARCHAR2,
i_day_part IN VARCHAR2,

i_default_month IN VARCHAR2,
i_default_day IN VARCHAR2,
i_yearmask_19 IN BOOLEAN)

RETURN OUT DATE

Action

Convert day, month, and year into a format that can be stored in Oracle as a date. This function is useful only if you store dates as multipart text items, which you may do to handle partial dates.

Arguments

This function uses the following arguments:

Argument:	Description:
i_year_part	Two-digit or four-digit string that represents the year, enclosed in single quotes. For example, '98' or '1998'. You should use a four-digit string to ensure Y2K compliance. This argument is required.
i_month_part	Two-digit string or text string that represents a month, enclosed in single quotes. For example, '03', 'MAR', or 'MARCH'. The default is Null.
i_day_part	Two-digit string that represents the day, enclosed in single quotes. For example, '21'. The default is Null.
i_default_month	Default month, enclosed in single quotes. For example: '12'. The default is '01'. This argument is used if i_month_part is Null.
i_default_day	Default day, enclosed in single quotes. For example: '23'. The default is '01'. This argument is used if i_day_part is Null.
i_yearmask_19	Applies only if a two-digit value i_year part is a two-digit string. If TRUE (the default), the year is stored as the 20th century. If FALSE, Oracle's default mask 'RR' is used, allowing you to store date values in other centuries. ***Note:*** The CONVERT_DATEI function is the same as CONVERT_DATE except that i_yearmask_19 is an integer.

Returned values

If a valid date is obtained, it is returned. Otherwise, it returns Null.

Example

Suppose that you store the birth date as three text items: B_MONTH, B_DAY, and B_YEAR, but you want to store the date as a date item, BIRTH_DATE. You could use the convert_date function as follows:

```
birth_date := ct_func.convert_date(this.b_year, this.b_month, this.b_day);
```

FIND_N_RECORDS

Function

```
FUNCTION FIND_N_RECORDS(
_protocol IN VARCHAR2,
i_panel IN VARCHAR2,
i_source IN VARCHAR2,
i_where_clause   IN VARCHAR2)

RETURN OUT NUMBER
```

Action

Determine the number of records matching a specified SQL WHERE clause.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	One of the following: • Clinical protocol. • Null when you search for a codelist. • Table owner when you search in an Oracle table not maintained by the Clintrial software.
i_panel	One of the following: • Clintrial software panel that contains the records to search. • Codelist name. • Table name of an Oracle table not maintained by the Clintrial software.
i_source	Type of table to search: • U — Clintrial software update table for a panel. • D — Clintrial software data table for a panel. • UD — Clintrial software update table, then the Clintrial software data table, if necessary. • DU — Clintrial software data table, then the Clintrial software update table, if necessary. • C — Clintrial software codelist table. • O — Other Oracle table, not maintained by the Clintrial software.
i_where_clause	SQL WHERE clause to use, or Null.

Returned values

Returns the number of values that match the specified condition. If no records are found, it returns 0.

Example

The following example determines the number of records in the HEMATOLOGY panel for the current subject:

```
rec_nbr :=  ct_func.find_n_records(ct_global.cts$protocol,
'HEMATOLOGY','UD',('sid =  ' ||this.sid));
```

The following example determines the number of records in the RACE codelist:

```
code_nbr := ct_func.find_n_records(NULL,'RACE','C',NULL);
```

IS_EMPTY

Function

```
FUNCTION IS_EMPTY(
i_item_values IN VARCHAR2)

RETURN OUT BOOLEAN
```

Note: The previous version of this function included the arguments *i_item_names* and *i_item_values*. The *i_item_names* argument is unnecessary. However, to maintain compatibility with your existing customized functions and procedures, the previous version of the function is available in addition to this new version.

Action

Check a list of items to determine if they are empty. An item is empty if its value is NULL, has zero length, or is a string containing only spaces.

Argument

This function uses the following argument:

Argument:	**Description:**
i_item_values	Delimiter-separated list of item values. Use the CT_STRING.MAKE_LIST function to create the list of item values.

Returned values

If all item values in the list of item values to check are empty, returns TRUE. Otherwise returns FALSE.

Example

The following example determines if the TEMP and PULSE items are empty:

```
ct_func.is_empty(ct_string.make_list(this.temp,this.pulse))
```

IS_NOTEMPTY

Function

FUNCTION IS_NOTEMPTY(
i_item_values IN VARCHAR2)
RETURN OUT BOOLEAN

Note: The previous version of this function included the arguments *i_item_names* and *i_item_values*. The *i_item_names* argument is unnecessary. However, to maintain compatibility with your existing customized functions and procedures, the previous version of the function is available in addition to this new version.

Action

Check a list of items to determine if they are not empty.

Argument

This function uses the following argument:

Argument:	Description:
i_item_values	Delimiter-separated list of item values. Use the CT_STRING.MAKE_LIST function to create the list of item values.

Returned values

If all items in a list are not empty, returns TRUE. Otherwise returns FALSE.

Example

The following example checks that five items (FNAME, LNAME, DOB, SSN, and VISIT_DATE) are not empty:

```
ct_func.is_notempty(
ct_string.make_list(this.fname,this.lname,this.dob, this.ssn,this.visit_date))
```

LOOKUP_FLAG

Function

FUNCTION LOOKUP_FLAG(
i_protocol IN VARCHAR2,
i_panel IN VARCHAR2,
i_ctrecid IN VARCHAR2,
i_cat_name IN VARCHAR2,
i_flag_name IN VARCHAR2,
i_level IN VARCHAR2,
i_item_name IN VARCHAR2,
o_comment OUT VARCHAR2,
io_tagid IN OUT NUMBER)

RETURN OUT BOOLEAN

Action

Determines whether a specified flag exists.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	Protocol name.
i_panel	Panel name.
i_ctrecid	Value of the system item CT_RECID.
i_cat_name	Flag category.
i_flag_name	Flag name.
i_level	I — The flag is attached to an item. R — The flag is attached to a record. O — The flag is attached to an observation.
i_item_name	Item name, if i_level is I.
o_comment	Flag comment.
io_tagid	Tag ID number of the flag. If you do not pass in the io_tagid, you must pass in the i_cat_name and i_flag_name.

Returned values

If the flag exists, returns True, and the tag ID number of the flag and the flag comment.

If the flag does not exist, returns False.

Example

The following example uses the LOOKUP_FLAG function:

```
// presume the user wants to determine if a flag
// exists on an item in the VITAL panel of the
// MEDIKA_CLINICAL protocol

o_comment VARCHAR2(100);
io_tagid  INTEGER;
b_flag_exists boolean;

b_flag_exists :=  ct_func.lookup_flag('MEDIKA_CLINICAL','VITAL',
    ct_global.cts$ct_recid,'VERIFICATION','AUTOFLAG','I',
'PULSE',o_comment,io_tagid );
```


LOOKUP_VARS

Function

FUNCTION LOOKUP_VARS(
i_protocol IN VARCHAR2,
i_panel IN VARCHAR2,
i_source IN VARCHAR2,
i_where IN VARCHAR2,
i_key_list IN VARCHAR2,
i_key_values IN VARCHAR2,
i_fetch_list IN VARCHAR2,
o_value_list OUT VARCHAR2,
io_lookup_id IN OUT NUMBER)

RETURN OUT VARCHAR2

Note: Previous versions of this function did not include the i_key_list, i_key_values, and io_lookup_id arguments, and they should be used if those arguments are not needed.

Action

Look up values for a list of items in specified tables, and place the values in a string.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	One of the following: • Clinical protocol. • Null to search a codelist. • Table owner when you search in an Oracle table not maintained by the Clintrial software.
i_panel	One of the following: • Clintrial software panel. • Codelist name. • Name of an Oracle table not maintained by the Clintrial software.
i_source	Table designator for the Clintrial software table to search: • U — Update table of a panel only. • D — Data table of a panel only. • UD — Update table, then the data table, if necessary. • DU — Data table, then the update table, if necessary. • C — Clintrial software codelist table. • O — Other Oracle table, not maintained by the Clintrial software.
i_where	SQL WHERE clause to use when looking up records, or Null.
i_key_list	Comma-separated list of key items, with each item name followed by an operator. For example: 'SUBJECT=, VISNO='
i_key_values	Comma-separated list of key values. For example: '120,2'
i_fetch_list	Comma-separated list of items. For example: 'SEX, RACE'
o_value_list	A delimiter-separated list of values for the items. (You can enter a variable that represents the list of values.) You can use the CT_STRING.GET_ITEM function to retrieve a value from o_value_list.

Argument:	Description:
io_lookup_id	For the first call with a particular combination of the protocol, panel, and source (the i_where, i_key_list, and i_fetch_list arguments), pass in Null as the io_lookup_id argument. The function will return a lookup number. For subsequent calls with the same combination of the protocol, panel, and source, pass in the lookup number as the io_lookup_id; this will cause the cursors to be reused, resulting in better performance.

Returned values

Returns a string containing a delimiter-separated list of items. Also returns the table designator and lookup number.

Example

The following example first looks up the TEST_NAME, UNITS, and RESULT items in the LABS_UPDATE table, and then in the LABS _DATA table, if necessary. In this example, the variable value_list contains the delimiter-separated list of values for the items.

```
tbl_found := ct_func.lookup_vars(ct_global.cts$protocol, 'LABS','UD',('TEST_NAME
= WBC'),'SUBJECT=,VISNO=','120,2'
'TEST_NAME,UNITS,RESULT',o_value_list,io_lookup_id);
```

To get the value of the RESULT from the o_value_list, you can use the CT_STRING.GET_ITEM function:

```
result_value := ct_string.get_item(o_value_list,3);
```

MSG_IF_EMPTY

Function

FUNCTION MSG_IF_EMPTY(
i_item_names IN VARCHAR2,
i_item_values IN VARCHAR2)
RETURN OUT VARCHAR2

Action

Check a list of items to determine if they are empty, and display a message if any items are empty. An item is empty if its value is Null, has zero length, or is a string containing only spaces.

Arguments

This function uses the following arguments:

Argument:	Description:
i_item_names	Comma-separated list of item names, enclosed in single quotes.
i_item_values	Delimiter-separated list of item values. You can use the CT_STRING.MAKE_LIST function to create the list of item values.

Returned values

If an item is empty, returns the following error message that contains a list of missing items

Missing values for items: *item1,item2,item3*

If an item is not empty, returns Null.

Example

The following example determines if values exist for the TEMP and PULSE items:

```
temp_pulse$msg := ct_func.msg_if_empty('TEMP,PULSE',
ct_string.make_list(this.temp,this.pulse));
```

If no value for the PULSE item exists, the following error message is displayed:

Missing values for items: PULSE

MSG_IF_NOTEMPTY

Function

FUNCTION MSG_IF_NOTEMPTY(
i_item_names IN VARCHAR2,
i_item_values IN VARCHAR2)
RETURN OUT VARCHAR2

Action

Check a list of items to determine if they are not empty.

Arguments

This function uses the following arguments:

Argument:	Description:
i_item_names	Comma-separated list of item names, enclosed in single quotes.
i_item_values	Delimiter-separated list of item values. Use the CT_STRING.MAKE_LIST function to create the list of item values.

Returned values

If an item is not empty, returns the following error message that contains a list of items that are not empty:

The following items should be empty but have values: *item1, item2,item3*

If an item is empty, returns Null.

Example

The following example determines if three items (FNAME, LNAME, DOB) are not empty:

```
demog_empty$msg := ct_func.msg_if_notempty('FNAME,LNAME,
DOB',ct_string.make_list(this.fname,this.lname,this.dob));
```

If the DOB item is not empty, the following error message is displayed:

The following items should be empty but have values: DOB

PAGE_HAS_DATA

Function

```
FUNCTION PAGE_HAS_DATA(
i_protocol IN VARCHAR2,
i_block_key INVARCHAR2,
i_block_repeat_key IN VARCHAR2,
i_page_key IN VARCHAR2,
i_page_repeat_key IN VARCHAR2,
i_subject_id IN INTEGER)

RETURN OUT INTEGER
```

Action

Determine if a page has existing records.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	Name of the protocol.
i_block_key	Block key value.
i_block_repeat-_key	Block repeat key value.

Argument:	Description:
i_page_key	Page key value.
i_page_repeat-_key	Page repeat key value.
i_subject_id	Value of the system item SUBJECT_ID. For a derivation or rule, use the variable ct_global.cts$subject_id. If i_subject_id is Null, the function checks for any subjects.

Returned values

If the page does not have existing records, returns 0.

If the page has existing records, returns 1.

If the protocol does not exist, returns -1.

Example

The following example determines if a page has existing records:

```
ret_status := ct_func.page_has_data(ct_global.cts$protocol, ct_global.cts$block,
null,ct_global.cts$page,null, ct_global.cts$subject_id);
```

SECTION_HAS_DATA

Function

FUNCTION SECTION_HAS_DATA(
i_protocol IN VARCHAR2,
i_block_key IN VARCHAR2,
i_block_repeat_key IN VARCHAR2,
i_page_key IN VARCHAR2,
i_page_repeat_key IN VARCHAR2,
i_panel IN VARCHAR2,

i_subject_id IN INTEGER)

RETURN OUT INTEGER

Action

Determine if a specified page section has existing records in a specific panel, and the table type of table (update table, data table, or both) that contains the records.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	Name of the protocol.
i_block_key	Block key value.
i_block_repeat-_key	Block repeat key value.
i_page_key	Page key value.
i_page_repeat-_key	Page repeat key value.
i_panel	Panel name.
i_subject_id	Value of the system item SUBJECT_ID. For a derivation or rule, use the variable ct_global.cts$subject_id. If i_subject_id is Null, the function checks for any subjects.

Returned values

If the section does not have existing records, returns 0.

If the section has existing records in the update table, returns 1.

If the section has existing records in the data table, returns 2.

If the section has existing records in both the update table and the data table, returns 3.

If the protocol or panel does not exist, returns -1.

Example

The following example determines if a page section has existing records:

```
ret_status := ct_func.section_has_data(ct_global.cts$protocol,
ct_global.cts$block,ct_global.cts$block_repeat,
ct_global.cts$page,ct_global.cts$page_repeat,
ct_global.cts$panel,ct_global.cts$subject_id);
```


String functions

What is a string function?

A *string function* is a Clintrial software function or procedure that is in the CT_STRING package, which is delivered with the Clintrial software. The string functions and procedures are available for use in Clintrial software derivations and rules, and two of the functions (INIT_NAME_VALUE-_ARRAYS and GET_ARRAY_VALUE) are available for data-entry processing procedures.

Note: The Design interface for creating derivations and rules may list procedures that are not described in this guide. Procedures that are not described in this guide are intended for Clintrial software internal use only.

How to call a string function

To call a string function, preface the function name with CT_STRING (the package name). For example:

```
ct_string.make_list(this.temp, this.pulse)
```

Other data types

The CT_STRING package also include two special data types:

- NameList — A list of names. The i_items argument for functions in the CT_EVENT package has this data type.
- NameValueList — A list of name/value pairs. The i_keys argument for functions in the CT_EVENT package has this data type.

You can also use these data types for the io_list argument in the ADD_ELEMENT function in the CT_STRING function.

To declare variables of these data types and construct an empty list, enter the following:

```
myNameList ct_string.NameList := ct_string.NameList();
myValueList ct_string.NameValueList := ct_string.NameValueList();
```

ADD_ELEMENT (for names)

Function

PROCEDURE ADD_ELEMENT(
io_list IN OUT ct_string.NameList,
i_name IN VARCHAR2)

Action

Add a specified name to a list of names. This procedure is intended for adding to lists created by functions and procedures in the CT_EVENT package.

Arguments

This procedure uses the following arguments:

Argument:	Description:
io_list	List of one or more names.
i_name	Name.

Example

The following example uses the ADD_ELEMENT procedure:

```
// presume the user wants to enable the item named 'TEMPC' based
// on the value of the item 'TEMPF'

myNameList ct_string.NameList := ct_string.NameList();

if (i_colname = 'TEMPF') AND (i_colvalue IS NULL) then
  ct_string.add_element(myNameList,'TEMPC');
  ct_event.enable(myNameList); -- re-enable the TEMPC field

end if;
```

ADD_ELEMENT (for name/value pairs)

Function

PROCEDURE ADD_ELEMENT(
io_list IN OUT ct_string.NameValueList,
i_name IN VARCHAR2,
i_value IN VARCHAR2)

Action

Add the specified name to a list of names. This procedure is intended for adding to lists created by functions and procedures in the CT_EVENT package.

Arguments

This procedure uses the following arguments:

Argument:	Description:
io_list	List of one or more name/value pairs.
i_name	Name.

Argument:	Description:
i_value	Value of the name specified by i_name.

Example

The following example uses the ADD_ELEMENT procedure:

```
// presume the user wants to set the value of the
// item named 'WGTKG' based on the value of item 'WGTLBS'
// whose value is passed in as i_colvalue

x number(4);
myValueList ct_string.NameValueList := ct_string.NameValueList();

x := to_number(i_colvalue);
ct_string.add_element(myValueList,'WGTKG',to_char(x  * .454));
ct_event.set_item(myValueList);
```

CHAR_TO_DATE

Function

FUNCTION CHAR_TO_DATE(
i_char IN VARCHAR2)
RETURN OUT DATE

Action

Convert a text string to an Oracle datetime. This function interprets the input argument as a text string in the Clintrial software internal format for a date, which is YYYYMMDDHH24MISS.

Arguments

This function uses the following argument:

Argument:	Description:
i_char	Text string.

Returned value

Returns an Oracle datetime.

Example

The following example uses the CHAR_TO_DATE function:

```
bdate := ct_string.char_to_date('19521123000000');
```

CHAR_TO_DATETIME

Function

```
FUNCTION CHAR_TO_DATETIME(
i_char IN VARCHAR2)
RETURN OUT DATE
```

Action

Convert a text string to an Oracle datetime. This function interprets the input argument as a text string in the Clintrial software internal format for a datetime, which is YYYYMMDDHH24MISS.

Arguments

This function uses the following argument:

Argument:	Description:
i_char	Text string.

Returned value

Returns an Oracle datetime.

Example

The following example uses the CHAR_TO_DATETIME function:

```
bdate := ct_string.char_to_datetime('19620314121042');
```

CHAR_TO_FLOAT

Function

```
FUNCTION CHAR_TO_FLOAT(
i_char IN VARCHAR2)
RETURN OUT NUMBER
```

Action

Convert a text string to an Oracle floating point number. This function interprets the input argument as a text string in the Clintrial software internal format for a floating point number. The Clintrial software internal format for a floating point number uses a period (.) as the decimal separator, and has at least one digit on each side of the decimal separator.

Arguments

This function uses the following argument:

Argument:	Description:
i_char	Text string.

Returned value

Returns an Oracle floating point number.

Example

The following example uses the CHAR_TO_FLOAT function:

```
tempf := ct_string.char_to_float('98.6');
```

COUNT_LIST

Function

FUNCTION COUNT_LIST(
i_list_of_items IN VARCHAR2,
i_delimiter IN VARCHAR2)
RETURN OUT INTEGER

Action

Count the number of items in a list of items.

Arguments

This function uses the following arguments:

Argument:	Description:
i_list_of_items	Delimiter-separated list of items, enclosed in single quotes.
i_delimiter	Character string to use as a delimiter. The default is '~*~'.

Returned values

Returns the number of items in the list. If the list is NULL, returns 0.

Example

The following example determines the number of items in a list:

```
nitems := ct_string.count_list('1,2,3,4,5',',');
```

The following example determines how many items are in a list separated by the '~' character:

```
nitems := ct_string.count_list('A~2~BC3~DE','~');
```

DATE_TO_CHAR

Function

```
FUNCTION DATE_TO_CHAR(
i_date IN DATE)
RETURN OUT VARCHAR2
```

Action

Convert an Oracle datetime to a text string. The returned value is a text string in the Clintrial software internal format for a date, which is YYYYMMDDHH24MISS.

Arguments

This function uses the following argument:

Argument:	Description:
i_date	Datetime.

Returned value

Returns a text string in the Clintrial software internal format for a date.

Example

The following example uses the DATE_TO_CHAR function:

```
txt_str := ct_string.date_to_char(birth_date);
```

If the birth_date variable has a value that means March 14, 1981, then the returned value is '19810314000000'.

DATETIME_TO_CHAR

Function

FUNCTION DATETIME_TO_CHAR(
i_date IN DATE)
RETURN OUT VARCHAR2

Action

Convert an Oracle datetime to a text string. The returned value is a text string in the Clintrial software internal format for a datetime, which is YYYYMMDDHH24MISS.

Arguments

This function uses the following argument:

Argument:	Description:
i_date	Datetime.

Returned value

Returns a text string in the Clintrial software internal format for a datetime.

Example

The following example uses the DATETIME_TO_CHAR function:

```
txt_string := ct_string.datetime_to_char(test_datetime);
```

If the test_datetime variable has a value that means 2:12:17 P.M. on May 23, 1999, then the returned value is '19990523141217'.

FLOAT_TO_CHAR

Function

FUNCTION FLOAT_TO_CHAR(
i_float IN NUMBER)
RETURN OUT VARCHAR2

Action

Convert an Oracle floating point number to a text string. This function interprets the input argument using the floating point number format for the Oracle server on which the function runs. The returned value is a text string in the Clintrial software internal format for a datetime. (the Clintrial software internal format for a floating point number uses a period (.) as the decimal separator, and has at least one digit on each side of the decimal separator.)

Arguments

This function uses the following argument:

Argument:	Description:
i_float	Number

Returned value

Returns a text string in the Clintrial software internal textual format for a floating point number.

Example

The following example uses the FLOAT_TO_CHAR function:

```
txt_string := ct_string.float_to_char(tempf);
```

If the tempf variable has a value that means 98.6, then the returned value is '98.6'.

GET_ARRAY_VALUE

Function

FUNCTION GET_ARRAY_VALUE(
i_item IN VARCHAR2)

RETURN OUT VARCHAR2

Action

Retrieve the value of a specified item from the array that is returned by the CT_STRING.INIT_NAME_VALUE_ARRAYS procedure.

Arguments

This function uses the following argument:

Argument:	Description:
i_item	Name of an item.

Returned value

Returns the value of the specified item.

Example

The following example uses the GET_ARRAY_VALUE function:

```
// presume the user wants to get the value of the item
// 'VISNO'. i_keys is an input parameter

x varchar2(20);

ct_string.init_name_value_arrays(i_keys);
x := ct_string.get_array_value('VISNO');
visitno := to_number(x);
```

GET_ITEM

Function

FUNCTION GET_ITEM(
i_list_of_items IN VARCHAR2,
i_number IN INTEGER)

RETURN OUT VARCHAR2

Action

Retrieve a specified item from a list of items.You can use this function to retrieve items from lists of items that were created by the CT_STRING.MAKE_LIST function or the CT_FUNC.LOOKUP_VARS function.

Arguments

This function uses the following arguments:

Argument:	Description:
i_list_of_items	Delimiter-separated list of items.
i_number	Number of the list element to return.

Returned value

Returns a particular item from the list, or NULL.

Example

The following example retrieves the third item from a list of item values. The list of item values is built by the CT_STRING.MAKE_LIST function and stored in the item_values variable:

```
one_value := ct_string.get_item(item_values,3)
```

For an example of using the CT_STRING.GET_ITEM function with the CT_FUNC.LOOKUP_VARS function, see page 320.

INIT_NAME_VALUE_ARRAYS

Function

FUNCTION INIT_NAME_ VALUE_ARRAYS(
i_items IN VARCHAR2)

Action

Create an array to store the values of a specified string of items.

Arguments

This function uses the following argument:

Argument:	Description:
i_items	Comma-separated list of item names and values, enclosed in single quotes.

Example

The following example uses the INIT_NAME_VALUE_ARRAYS function:

```
// presume the user wants to get the value of the item
// 'VISNO'. i_keys is an input parameter

x varchar2(20);

ct_string.init_name_value_arrays(i_keys);
x := ct_string.get_array_value('VISNO');
visitno := to_number(x);
```

MAKE_LIST

Function

FUNCTION MAKE_LIST (

i_elm1 IN VARCHAR2,

i_elm2 IN VARCHAR2,

i_elm3 IN VARCHAR2,

i_elm4 IN VARCHAR2,

i_elm5 IN VARCHAR2,

i_elm6 IN VARCHAR2,

i_elm7 IN VARCHAR2,

i_elm8 IN VARCHAR2,

i_elm9 IN VARCHAR2,

i_elm10 IN VARCHAR2,

i_elm11 IN VARCHAR2,

i_elm12 IN VARCHAR2,

i_elm13 IN VARCHAR2,

i_elm14 IN VARCHAR2,

i_elm15 IN VARCHAR2,

i_elm16 IN VARCHAR2,

i_elm17 IN VARCHAR2,

i_elm18 IN VARCHAR2,

i_elm19 IN VARCHAR2,

i_elm20 IN VARCHAR2)

RETURN OUT VARCHAR2

Action

Create a list of up to 20 items that are separated by a constant delimiter (‘~*~’).

Arguments

This function uses the following argument:

Arguments	**Description:**
i_elm1 to i_elm20	Up to 20 items to use in creating the list. This function is most often used to create a list of Clintrial software item values. The default is ‘!&@^#%$’.

Returned values

Returns a string containing a concatenated list of items using a constant delimiter ‘~*~’.

Example

The following example creates a concatenated list of two items, TEMP and PULSE:

```
vallist:= ct_string.make_list(this.temp,this.pulse));
```

Privilege functions

What is a privilege function?

A *privilege function* is a Clintrial software function that is in the CT_PROC_ACCOUNT package, which is delivered with the Clintrial software.

How to call a privilege function

To call a privilege function, preface the function name with CT_PROC_ACCOUNT (the package name).

The only available privilege function is GRANT_EXECUTE_PRIVS. For information about this function, see page 301.

Resolve functions

What is a Resolve function?

A *Resolve function* is a Clintrial software function that is specifically related to Resolve. Resolve functions are available for use in Clintrial software derivations and rules (but not data-entry processing procedures). The Resolve functions are in the CTV_CORE package, which is created automatically when you install Resolve. (The CTV_CORE package also contains other CTV_CORE functions, which are used internally by Resolve and should not be called by derivations and rules.)

How to call a Resolve function

To call a Resolve function, preface the function name with CTV_CORE (the package name).

For information about the Resolve functions, see the *Resolve* section of *Enter, Resolve, and Retrieve*. The Resolve functions include:

- FLAG_TO_DISCREPANCY

- INV_ID
- SETUP_ERROR_ITEM

Lab Loader functions

What is a Lab Loader function?

A *Lab Loader function* is a Clintrial software function that is specifically related to Lab Loader. Lab Loader functions are available for use in Clintrial software derivations and rules (but not data-entry processing procedures). The Lab Loader functions are in the CTL_FUNC package, which is created automatically when you install Lab Loader (the CTL_FUNC package also contains other functions, which are used internally by Lab Loader and should not be called by derivations and rules).

How to call a Lab Loader function

To call a Lab Loader function, preface the function name with CTL_FUNC (the package name). For example:

```
ctl_func.calc_normal_status(this.labid,this.testcode, this.testdt, this.testvalues);
```

Note: The LOOKUP_SUBJECT_ID function is installed as part of the CT_FUNC package.

CALC_NORMALCY_STATUS

Function

```
FUNCTION CALC_NORMALCY_STATUS(
i_labid IN VARCHAR2,
i_testcode IN VARCHAR2,
i_testdt IN DATE,
i_testvalue IN VARCHAR2,
i_protocol IN VARCHAR2,
i_dmg_panel IN VARCHAR2,
```

i_subject IN INTEGER,
i_sex_item IN VARCHAR2,
i_age_item IN VARCHAR2,
i_age_unit IN VARCHAR2,
i_wt_item IN VARCHAR2,
i_wt_unit IN VARCHAR2)
RETURN OUT VARCHAR2

Action

Calculates the normalcy status of a lab result value based on the normal range of values. The low and high range is calculated by calling CALC_NORMAL_RANGE.

Arguments

This function uses the following arguments:

Argument:	Description:
i_labid	Unique record identifier (required).
i_testcode	Lab test code (required).
i_testdt	Date of lab test (required).
i_testvalue	Value of lab test result (required).
i_protocol	Protocol name.
i_dmg_panel	Name of the panel with the demographic information.
i_subject	The record's subject_id.
i_sex_item	Name of item where sex is stored.
i_age_item	Name of item where age is stored.
i_age_unit	Name of item where age units are stored.
i_wt_item	Name of item where weight is stored.

Argument:	Description:
i_wt_unit	Name of item where weight units are stored.

Note: Required arguments must have non-null values. The i_protocol, i_dmg_panel, and i_subject arguments must be non-null if the lab normal ranges depend on sex, age or weight.

Returned value

Returns an H (above high normal), L (below low normal), N (normal) or null (no range defined, or normal ranges do not apply).

Example

The following example calculates the normalcy status of the value of a lab test result:

```
normalcy_status := ctl_func.calc_normalcy_status(LAB_ID,TEST_CODE,
TEST_DATE, TEST_RESULT,);
```


CALC_NORMAL_RANGE

Function

```
FUNCTION CALC_NORMAL_RANGE(
i_labid IN VARCHAR2,
i_testcode IN VARCHAR2,
i_testdt IN DATE,
i_testvalue IN VARCHAR2,
i_protocol IN VARCHAR2,
i_dmg_panel IN VARCHAR2,
i_subject_ IN INTEGER,
i_sex_item IN VARCHAR2,
i_age_item IN VARCHAR2,
i_age_unit IN VARCHAR2,
i_wt_item IN VARCHAR2,
i_wt_unit IN VARCHAR2)
```

o_low_range IN VARCHAR2,
o_high_range IN VARCHAR2)
RETURN OUT VARCHAR2

Action

Calculates the low and high normal ranges for a series of lab test values.

Arguments

This function uses the following arguments:

Argument:	Description:
i_labid	Unique record identifier (required).
i_testcode	Lab test code (required).
i_testdt	Date of lab test (required).
i_testvalue	Value of lab test result (required).
i_protocol	Protocol name.
i_dmg_panel	Name of the panel with the demographic information.
i_subject	The record's subject_id.
i_sex_item	Name of item where sex is stored.
i_age_item	Name of item where age is stored.
i_age_unit	Name of item where age units are stored.
i_wt_item	Name of item where weight is stored.
i_wt_unit	Name of item where weight units are stored.
o_low_range	Low normal value.
o_high_range	High normal value.

Note: Required arguments must have non-null values. The i_protocol, i_dmg_panel, and i_subject arguments must be non-null if the lab normal ranges depend on sex, age or weight.

Returned value

Returns the high and low normal ranges.

Example

The following example calculates the low and high normal ranges for a lab test:

```
normalcy_range := ctl_func.calc_normal_range(LAB_ID,TEST_CODE, TEST_DATE,
TEST_RESULT,);
```


CALC_SI_VALUE

Function

FUNCTION CALC_SI_VALUE(
i_src_unit IN VARCHAR2,
i_dest_unit IN VARCHAR2,
i_testvalue IN NUMBER,
i_testcode IN VARCHAR2)
RETURN OUT NUMBER

Action

Calculates the value of a lab test result in SI units.

Arguments

This function uses the following arguments:

Argument:	Description:
i_src_unit	Lab unit to be converted from (required).
i_dest_unit	Lab unit to be converted to (required).
i_testvalue	Lab test result value to be converted (required).
i_testcode	Value of lab test result.

Note: Required arguments must have non-null values.

Returned value

Returns the value of the lab result in SI units.

Example

The following example calculates the result of a lab test in SI units:

```
si_value := ctl_func.calc_si_value(LAB_UNITS,SI_UNITS, TEST_VALUE);
```

LOOKUP_SUBJECT_ID

Function

FUNCTION LOOKUP_SUBJECT_ID(
i_protocol IN VARCHAR2,
i_subject IN VARCHAR2)
RETURN OUT INTEGER

Note: This function is installed as part of the CT_FUNC package.

Action

Looks up a patient's SUBJECT_ID.

Arguments

This function uses the following arguments:

Argument:	Description:
i_protocol	Protocol name.
i_subject	Subject item name.

Returned value

Returns the SUBJECT_ID of a patient.

Example

The following example finds the SUBJECT_ID of a patient:

```
patient_subject_id := ct_func.lookup_subject_id('MEDIKA_CLINICAL',
PATIENT_ID);
```


MedDRA functions

What is a MedDRA function?

A *MedDRA function* is a Clintrial software function that is specifically related to the CT_MEDDRA thesaurus protocol. The Clintrial software provides MedDRA-related functions to:

- Create indexes on panels in the CT_MEDDRA protocol supplied by the Clintrial software
- Use in SELECT statements through SQL Tools or in derivations

- Use in Retrieve queries supplied by the Clintrial software that query clinical data coded to the MedDRA dictionary

The MedDRA functions are contained within the thesaurus protocol created for Meddra. These functions should be created during the setup of the Meddra thesaurus protocol. For information on setting up these functions for a sample thesaurus protocol named CT_MEDDRA, see the Clintrial 4.7 *Admin & Design* manual, **Chapter 14: Coding Thesauruses** in the section entitled, "Setting up CT_MEDDRA protocol."

Note: The source code is open and any of the functions below could be altered to accept the names, rather than the codes as input parameters.

How to call a MedDRA function

To call a MedDRA function, preface the function name with CT_MEDDRA (the package name). For example:

```
ct_meddra.get_code_llt(this.i_llt, this.i_return_type);
```


GET_CODE_LLT

Function

FUNCTION GET_CODE_LLT(
i_llt IN VARCHAR2,
i_return_type IN INTEGER)
RETURN OUT NUMBER

Action

Retrieves the code associated with a low level term code.

Arguments

This function uses the following arguments:

Argument:	Description:
i_llt	Low level term code (required).
i_return_type	Type of MedDRA code: 1 — system organ class 2 — high level group term 3 — high level term 4 — preferred term (default) 5 — low level term

Returned value

Returns the code associated with a specified low level term code.

Example

The following example finds the code for a specified low level term:

```
meddra_LLT_code := ct_meddra.get_code_llt(LLT_CODE, 4);
```

GET_CODE_PT

Function

FUNCTION GET_CODE_PT(
i_pt IN VARCHAR2,
i_return_type IN INTEGER)
RETURN OUT NUMBER

Action

Retrieves the code associated with a preferred term code.

Arguments

This function uses the following arguments:

Argument:	Description:
i_pt	Preferred term code (required).
i_return_type	Type of MedDRA code: 1 — system organ class 2 — high level group term 3 — high level term 4 — preferred term (default) 5 — low level term

Returned value

Returns the code associated with a specified preferred term code.

Example

The following example finds the code for a specified preferred term:

```
meddra_PT_code := ct_meddra.get_code_pt(PT_CODE, 4);
```

GET_CODE_HLT

Function

FUNCTION GET_CODE_HLT(
i_hlt IN VARCHAR2,
i_return_type IN INTEGER)
RETURN OUT NUMBER

Action

Retrieves the code associated with a high level term code.

Arguments

This function uses the following arguments:

Argument:	Description:
i_hlt	High level term code (required).
i_return_type	Type of MedDRA code: 1 — system organ class 2 — high level group term 3 — high level term 4 — preferred term (default) 5 — low level term

Returned value

Returns the code associated with a specified high level term code.

Example

The following example finds the code for a specified high level term:

```
meddra_HLT_code := ct_meddra.get_code_hlt(HLT_CODE, 4);
```

GET_CODE_HLGT

Function

FUNCTION GET_CODE_HLGT(
i_hlgt IN VARCHAR2,
i_return_type IN INTEGER)
RETURN OUT NUMBER

Action

Retrieves the code associated with a high level group term code.

Arguments

This function uses the following arguments:

Argument:	Description:
i_hlgt	High level group term code (required).
i_return_type	Type of MedDRA code: 1 — system organ class 2 — high level group term 3 — high level term 4 — preferred term (default) 5 — low level term

Returned value

Returns the code associated with a specified high level group term code.

Example

The following example finds the code for a specified high level group term:

```
meddra_HLGT_code := ct_meddra.get_code_hlgt(HLGT_CODE, 4);
```

GET_TERM

Function

FUNCTION GET_TERM(
i_code IN INTEGER,
i_input_type IN INTEGER)
RETURN OUT VARCHAR2

Action

Retrieves the name associated with a low level code.

Arguments

This function uses the following arguments:

Argument:	Description:
i_code	Code (required).
i_input_type	Type of MedDRA code: 1 — system organ class 2 — high level group term 3 — high level term 4 — preferred term 5 — low level term

Returned value

Returns the name associated with a MedDRA code.

Example

The following example finds the low level term name for a specified low level term code:

meddra_LLT_name := ct_meddra.get_term('LLT_CODE', 5);

Event utility functions

What is an event utility function?

An *event utility function* is a Clintrial software function or procedure that is in the CT_EVENT package, which is delivered with the Clintrial software. The event utility functions are available for use in Clintrial software data-entry processing procedures (but not in derivations or rules).

How to call an event utility function

To call an event utility function, preface the function name with CT_EVENT (the package name). For example, use the following PL/SQL:

ct_event.ENABLE

DELETE_RPT

Procedure

PROCEDURE DELETE_RPT(
i_keys IN ct_string.NameValueList,
i_delete_reason IN VARCHAR2)

Action

Applies only to the Initializing Page Section and Value Changed events.

Delete the records (and associated tags and discrepancies) identified by the specific keys if the row is not protected. A row can be protected by user security, panel security, or the CT_EVENT.DISABLE_DEL function.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Note: To use DELETE_RPT in Verify, remove preset sequences from the page section, and use ct_string.add_element(), ct_event.set_rpt(), ct_event.enable_del(); ct_event.delete_rpt().

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_delete_reason	Reason for the deletion.

DEL_FLAG

Procedure

PROCEDURE DEL_FLAG(
i_category_name IN VARCHAR2,
i_flag_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2)

Action

Applies only to the Initializing Page Section and Value Changed events.

Delete a flag that meets the specified criteria.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_category_name	Flag category.
i_flag_name	Flag name.
i_aggregation	I — The flag is attached to an item. R — The flag is attached to a record. O — The flag is attached to an observation.
i_item	Name of an item, if i_aggregation is I.

Example

The following examples use the DEL_FLAG procedure:

```
// presume that the user wants to remove the 'GENERAL'
// 'DELTA_BP' flag that has been attached
// to the item 'BPDIA'

ct_event.del_flag('GENERAL','DELTA_BP','I', 'BPDIA');
```

DEL_FLAG_RPT

Procedure

PROCEDURE DEL_FLAG_RPT(
i_keys IN ct_string.NameValueList,
i_category_name IN VARCHAR2,
i_flag_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2)

Action

Applies only to the Initializing Page Section and Value Changed events.

Delete a flag that meets the specified criteria from records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	**Description:**
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_category-_name	Flag category.
i_flag_name	Flag name.
i_aggregation	I — The flag is attached to an item. R — The flag is attached to a record. O — The flag is attached to an observation.
i_item	Name of an item, if i_aggregation is I.

Example

The following example uses the DEL_FLAG_RPT procedure:

```
// presume that the user wants to remove the 'VERIFICATION'
// 'AUTOFLAG' flag that has been attached
// to the repeating item 'PULSE'. i_itemvalues is an
// input parameter

ct_string.init_name_value_arrays(i_itemvalues);
ct_string.add_element(KeyNameValue, 'PID',
ct_string.get_array_value('PID'));
ct_string.add_element(KeyNameValue, 'VISIT',
ct_string.get_array_value('VISIT'));
ct_string.add_element(KeyNameValue, 'VISRPT',
ct_string.get_array_value('VISRPT'));
ct_event.DEL_FLAG_RPT(KeyNameValue,'VERIFICATION','AUTOFLAG',
  'I', 'PULSE');
```

DEL_NOTE

Procedure

PROCEDURE DEL_NOTE(
i_category_name IN VARCHAR2,
i_note_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2)

Action

Applies only to the Initializing Page Section and Value Changed events.

Delete a note that meets the specified criteria.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_category_name	Note category.
i_note_name	Note name.

Argument:	Description:
i_aggregation	I — The note is attached to an item. R — The note is attached to a record. O — The note is attached to an observation.
i_item	Name of an item, if i_aggregation is I.

Example

The following example uses the DEL_NOTE procedure:

```
// presume the user wants to delete a
// note attached to the item PULSE

ct_event.DEL_NOTE('VERIFICATION','AUTOFLAG','I','PULSE') ;
```


DEL_NOTE_RPT

Procedure

PROCEDURE DEL_NOTE_RPT(
i_keys IN ct_string.NameValueList,
i_category_name IN VARCHAR2,
i_note_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2)

Action

Applies only to the Initializing Page Section and Value Changed events.

Delete a note that meets the specified criteria from records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_category-_name	Note category.
i_note_name	Note name.
i_aggregation	I — The note is attached to an item. R — The note is attached to a record. O — The note is attached to an observation.
i_item	Name of an item, if i_aggregation is I.

DISABLE

Procedure

PROCEDURE DISABLE(
i_items IN ct_string.NameList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Prevent (disable) editing of the specified items. If used from a repeating page section, this procedure applies to the current row.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_items	List of one or more item names.

Example

The following example uses the DISABLE function:

```
// presume the user wants to disable input to an item named 'ABNCOMM'
// based on the value of another item. This example could be coded in //thevalue changed
event (where i_colvalue is an input param)

myNameList ct_string.NameList := ct_string.NameList();

if (i_colvalue IS NOT NULL) then
  if i_colvalue = 1 then
    ct_string.add_element(myNameList,'ABNCOMM');
    ct_event.disable(myNameList);
  end if;

end if;
```


DISABLE_DEL

Procedure

PROCEDURE DISABLE_DEL(
i_keys IN ct_string.NameValueList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Prevent (disable) deletion of records identified by the specified keys.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.

Example

The following example uses the DISABLE_DEL procedure:

```
// presume the user wants to disable deletion of
// the item PID

KeyNameValue ct_string.NameValueList := ct_string.NameValueList();

ct_string.init_name_value_arrays(i_itemvalues);
ct_string.add_element(KeyNameValue, 'PID',ct_string.get_array_value('PID'));
ct_string.add_element(KeyNameValue, 'VISIT', ct_string.get_array_value('VISIT'));

ct_event.DISABLE_DEL(KeyNameValue);
```

DISABLE_RPT

Procedure

PROCEDURE DISABLE_RPT(
i_keys IN ct_string.NameValueList,
i_items IN ct_string.NameList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Prevent (disable) editing of the specified items for records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_items	List of one or more item names.

Example

The following example uses the DISABLE_RPT procedure:

```
// presume the user wants to disable editing
// of the the repeating item TEMPC

KeyNameValue ct_string.NameValueList := ct_string.NameValueList();
myNameList   ct_string.NameList := ct_string.NameList();

ct_string.init_name_value_arrays(i_itemvalues);

ct_string.add_element(KeyNameValue, 'PID',
ct_string.get_array_value('PID'));
ct_string.add_element(KeyNameValue, 'VISIT', ct_string.get_array_value('VISIT'));
ct_string.add_element(KeyNameValue, 'VISRPT', ct_string.get_array_value('VISRPT'));
ct_string.add_element(myNameList,'TEMPC');

ct_event.DISABLE_RPT(KeyNameValue, myNameList) ;
```

ENABLE

Procedure

PROCEDURE ENABLE(
i_items IN ct_string.NameList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Allow (enable) editing of the specified items if the items were disabled by CT_EVENT.DISABLE.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_items	List of one or more item names.

Example

The following example uses the ENABLE procedure:

```
// presume the user wants to enable input to an item named 'ABNCOMM'
// based on the value of another item. This example could be coded in
// the value changed event (where i_colvalue is an input param)

myNameList ct_string.NameList := ct_string.NameList();

if (i_colvalue IS NOT NULL) then
  if i_colvalue = 2 then
    ct_string.add_element(myNameList,'ABNCOMM');
    ct_event.enable(myNameList);
  end if;

end if;
```

ENABLE_DEL

Procedure

PROCEDURE ENABLE_DEL(
i_keys IN ct_string.NameValueList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Allow (enable) deletion of records identified by the specified keys.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.

Example

The following example uses the ENABLE_DEL procedure:

```
// presume the user wants to enable deletion of
// the item PID

KeyNameValue  ct_string.NameValueList := ct_string.NameValueList();

ct_string.init_name_value_arrays(i_itemvalues);
ct_string.add_element(KeyNameValue, 'PID',ct_string.get_array_value('PID'));

ct_event.ENABLE_DEL(KeyNameValue) ;
```

ENABLE_RPT

Procedure

PROCEDURE ENABLE_RPT(
i_keys IN ct_string.NameValueList,
i_items IN ct_string.NameList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Allow (enable) editing of the specified items for records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_items	List of one or more item names.

Example

The following example uses the ENABLE_RPT procedure:

```
// presume the user wants to enable editing
// of the the repeating item TEMPC
// i_itemvalues is an input parameter

KeyNameValue ct_string.NameValueList := ct_string.NameValueList();
myNameList   ct_string.NameList := ct_string.NameList();

ct_string.init_name_value_arrays(i_itemvalues);

ct_string.add_element(KeyNameValue, 'VISIT', ct_string.get_array_value('VISIT'));
ct_string.add_element(KeyNameValue, 'VISRPT', ct_string.get_array_value('VISRPT'));
ct_string.add_element(myNameList,'TEMPC');

ct_event.ENABLE_RPT(KeyNameValue, myNameList) ;
```

FLAG

Procedure

```
PROCEDURE FLAG(
i_category_name IN VARCHAR2,
i_flag_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2,
i_comment IN VARCHAR2,
i_append IN INTEGER)
```

Action

Applies only to the Initializing Page Section and Value Changed events.

Create or edit a flag for the current record.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_category-_name	Flag category. Flags in the VERIFICATION category can be added only during verification.
i_flag_name	Flag name; must be a flag name defined in Design. Flags with the name AUTOFLAG cannot be added with the function.
i_aggregation	I — The flag is attached to an item. R — The flag is attached to a record. O — The flag is attached to an observation.
i_item	Name of an item, if i_aggregation is I.
i_comment	Flag comment.
i_append	0 — Replace the existing comment; this is the default. 1 — Append the specified comment to the existing comment.

Example

The following example uses the FLAG procedure:

```
// presume the user wants to create a 'GENERAL' 'DELTA_BP'
// flag on the item 'BPDIA'

flag_text varchar2(240);

flag_text := 'Diastolic Delta-BP > 10% since last visit';
ct_event.flag('GENERAL', 'DELTA_BP', 'I','bpdia',flag_text,0);
```

FLAG_RPT

Procedure

PROCEDURE FLAG_RPT(
i_keys IN ct_string.NameValueList,
i_category_name IN VARCHAR2,
i_flag_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2,
i_comment IN VARCHAR2,
i_append IN INTEGER)

Action

Applies only to the Initializing Page Section and Value Changed events.

Create or edit a flag for records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	**Description:**
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_category_name	Flag category. Flags in the VERIFICATION category can be added only during verification.
i_flag_name	Flag name; must be a flag name defined in Design. Flags with the name AUTOFLAG cannot be added with the function.

Argument:	Description:
i_aggregation	I — The flag is attached to an item. R — The flag is attached to a record. O — The flag is attached to an observation.
i_item	Name of an item, if i_aggregation is I.
i_comment	Flag comment.
i_append	0 — Replace the existing comment; this is the default. 1 — Append the specified comment to the existing comment.

Example

The following example uses the FLAG_RPT procedure:

```
// presume the user wants to create a 'GENERAL' 'DELTA_BP'
// flag on the repeating item 'BPDIA'. i_keys is an input parameter

flag_text varchar2(240);

flag_text := 'Diastolic Delta-BP > 10% since last visit';
ct_event.flag_rpt(i_keys, 'GENERAL', 'DELTA_BP', 'I','bpdia',flag_text,0);
```

ITEM_FOCUS

Procedure

PROCEDURE ITEM_FOCUS(
i_item IN VARCHAR2)

Action

Applies only to the Value Changed event.

Moves the cursor to an item in the current record if the item is enterable. If the item is not enterable, the cursor does not move.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_item	One of the following: • Name of an item. • CTS$NEXT_SECTION - Place the cursor on the first enterable item in the first row of the next page section. • CTS$PREV_SECTION - Place the cursor on the first enterable item in the first row of the previous page section.

Example

The following example uses the ITEM_FOCUS procedure:

```
l_col_name :='RACEOTH';
l_other_col_name := 'ALLERG';
ct_string.add_element(myNameList,l_col_name);
ct_string.add_element(myOtherNameList,l_other_col_name);
if i_colvalue = '5'
then
 ct_event.enable(myNameList);
 ct_event.item_focus(l_col_name);
else
 ct_event.disable(myNameList);
 ct_event.item_focus(l_other_col_name);
end if;
o_result :=1;
END itemfocus;
```

ITEM_FOCUS_RPT

Procedure

PROCEDURE ITEM_FOCUS_RPT(
i_keys IN ct_string.NameValueList,
i_item IN VARCHAR2)

Action

Applies only to the Value Changed event.

Move the cursor to an item in the first record that matches the specified keys, if the item is enterable. If the item is not enterable, the cursor does not move.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	**Description:**
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_item	One of the following: • Name of an item • CTS$NEXT_SECTION — Place the cursor on the first enterable item in the first row of the next page section. • CTS$PREV_SECTION — Place the cursor on the first enterable item in the first row of the previous page section.

Example

The following example uses the ITEM_FOCUS_RPT procedure:

```
// presume the user wants to set focus to a repeating item TEMPC,
// i_itemvalues is an input parameter

KeyNameValue ct_string.NameValueList := ct_string.NameValueList();
myNameList ct_string.NameList := ct_string.NameList();

ct_string.init_name_value_arrays(i_itemvalues);

ct_string.add_element(KeyNameValue, 'VISIT', ct_string.get_array_value('VISIT'));
ct_string.add_element(KeyNameValue, 'VISRPT', ct_string.get_array_value('VISRPT'));
ct_string.add_element(myNameList,'TEMPC');

ct_event.ITEM_FOCUS_RPT(KeyNameValue, myNameList) ;
```

NOTE

Procedure

PROCEDURE NOTE(
i_category_name IN VARCHAR2,
i_note_name IN VARCHAR2,
i_aggregation IN VARCHAR2
i_item IN VARCHAR2
i_comment IN VARCHAR2,
i_append IN INTEGER)

Action

Applies only to the Initializing Page Section and Value Changed events.

Create or edit a note for the current record.

Arguments

This procedure uses the following arguments:

Argument:	Description:
i_category_name	Note category.
i_note_name	Note name; must be note name defined in Design.
i_aggregation	I — The note is attached to an item. R — The note is attached to a record. O — The note is attached to an observation.
i_item	Name of an item, if i_aggregation is I.
i_comment	Note comment.
i_append	0 — Replace the existing comment; this is the default. 1 — Append the specified comment to the existing comment.

Example

The following example uses the NOTE procedure:

```
// presume the user wants to create an 'INVESTIGATOR' 'UNSPECIFIED'
// note on the item 'BPDIA'

note_text varchar2(240);

note_text := 'Data illegible';
ct_event.note('INVESTIGATOR', 'UNSPECIFIED', 'I','bpdia',note_text,0);
```

NOTE_RPT

Procedure

PROCEDURE NOTE_RPT(
i_keys IN ct_string.NameValueList,
i_category_name IN VARCHAR2,
i_note_name IN VARCHAR2,
i_aggregation IN VARCHAR2,
i_item IN VARCHAR2,
i_comment IN VARCHAR2,
i_append IN INTEGER)

Action

Applies only to the Initializing Page Section and Value Changed events.

Create or edit a note for records identified by the specified keys.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	**Description:**
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_category_name	Note category.
i_note_name	Note name; must be a note name defined in Design.

Argument:	Description:
i_aggregation	I — The note is attached to an item. R — The note is attached to a record. O — The note is attached to an observation.
i_item	Name of an item, if i_aggregation is I.
i_comment	Note comment.
i_append	0 — Replace the existing comment; this is the default. 1 — Append the specified comment to the existing comment.

Example

The following example uses the NOTE_RPT procedure:

```
// presume the user wants to create a 'INVESTIGATOR', 'UNSPECIFIED'
// note on the repeating item 'BPDIA'. i_keys is an input parameter

note_text varchar2(240);

note_text := 'Data Illegible';
ct_event.note_rpt(i_keys, 'INVESTIGATOR', 'UNSPECIFIED,
  'I','bpdia',note_text,0);
```

SECTION_FOCUS

Procedure

PROCEDURE SECTION_FOCUS(
i_section_name IN VARCHAR2)

Action

Applies only to the Value Changed event.

Move the cursor to the first enterable item in the next occurrence of the specified page section. If there is no enterable item, the cursor is not moved.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_section_name	Name of the page section.

Example

The following example uses the SECTION_FOCUS procedure:

```
// presume the user wants to set focus to
// the section INCLUS

ct_event.SECTION_FOCUS ('INCLUS');
```

SET_ITEM

Procedure

PROCEDURE SET_ITEM(
i_item_values IN ct_string.NameValueList*)*

Action

Applies only to the Initializing Page Section and Value Changed events.

Set the specified items to the specified values, except that:

- Context items can be changed only if the current page section is the context page section. Changes to context items are propagated to each record on a page.

- The item cannot be a subject item, block key item, page key item, block repeat key item, page repeat key item, a subset page section key item, master key item, or detail key item.

When used in a repeating page section, this procedure applies to the current row.

When used in the non-repeating page section of a within-panel master-detail relationship, then changes are propagated to the associated repeating page section when the page is saved.

Arguments

This procedure uses the following argument:

Argument:	Description:
i_item_values	List of item/value pairs. Use the CT_STRING.ADD_ELEMENT function to populate the list.

SET_RPT

Procedure

PROCEDURE SET_RPT(
i_keys IN ct_string.NameValueList,
i_item_values IN ct_string.NameValueList)

Action

Applies only to the Initializing Page Section and Value Changed events.

Set the specified items to the specified values for records identified by the specified keys, except that:

- Context items cannot be changed.
- The item cannot be a subject item, block key item, page key item, block repeat key item, page repeat key item, a subset page section key item, master key item, or detail key item.

If the panel is the detail of a cross-panel master-detail relationship, then a record with the specified keys must exist in the master panel.

If there are no existing records with the specified keys, then records are created.

If the panel is the detail of master panel that is repeating, then the master key must be included in the i_keys argument.

Note: If this procedure is used for a non-repeating page section, an error occurs.

Arguments

This procedure uses the following arguments:

Argument:	**Description:**
i_keys	List of key item name and item value pairs that may include the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. If fewer keys are specified, then the procedure acts on all rows with the specified keys.
i_item_values	List of item and value pairs. Use the CT_STRING.ADD_ELEMENT function to populate the list.

17 Using Data-Entry Processing Procedures

Overview

What is a data-entry processing procedure?

A *data-entry processing procedure* (DEPP) is a PL/SQL procedure that runs when the Enter user performs a particular action related to a study page, page section, or item.

For example, a data-entry processing procedure attached to an item can convert an entered value to another value, or move the cursor to a particular field based on the value entered in the current field. Or, a data-entry processing procedure attached to a page section could carry forward values from the previous page section.

You write data-entry processing procedures in PL/SQL, and attach the procedures to page templates, page sections, or items in Design. Data-entry procedures must be set up outside of Design, and called from within it. This differs from derivations and rules, which can be either set up outside of Design and called from within it, or set up within Design.

What is a data processing event?

A *data processing event* is a Clintrial software action to which a data-entry processing procedure can be attached. The following table lists the data processing events:

Event:	This event occurs when:
Page Opened	The user opens the study page.
Page Saved	The user has saved data on the study page. The data-entry processing procedure runs *after* changes are committed to the database.
Page Deleted	The user has deleted data on the study page. The data-entry processing procedure runs *after* changes are committed to the database.

Event:	This event occurs when:
Initializing Page Section	The user opens a study page. ***Note:*** Even though this procedure is related to a page section, it runs when the study page is opened. If there are also Page Opened procedures, then they run before the Initializing Page Section procedures.
Saving Page Section	The user saves data on the study page. The data-entry processing procedure runs *before* changes are committed to the database. ***Note:*** Even though this procedure is related to a page section, it runs when the study page is saved. If there is also a Page Saved procedure, then the Saving Page Section procedures run before the Page Saved procedure.
Value Changed	The user changes the value of an item on a study page and leaves the field. The data-entry processing procedure runs *before* changes are committed to the database. For more detail about when a Value Changed procedure is run, see the next section. ***Note*:** Value Changed procedures are an enhancement of conversion procedures, which were available in previous releases of Clintrial software. Your existing conversion procedures from prior releases will work in the Clintrial software.

Value Changed procedures

When procedures are run

The following steps explain when a Value Changed procedure is run in relation to other types of checks performed on an entered value:

1. The user enters a value in the field, by typing it in, selecting a value from a codelist (or checklist), or changing the state of a radio button or check box. At this point, the following checks are performed:
   - For a text field, the length of the value cannot be longer than the length specified by the item's database format.
   - For numeric fields, the value must be digits, +, -, or the appropriate decimal point character. If appropriate, a decimal point is inserted automatically. If a decimal grouping character is entered, it is ignored.

The decimal grouping character is a comma in American notation and a period in European notation. The decimal point character is a period in American notation and a comma in European notation. The length of the value cannot be longer than the length specified by the item's database format.

- For date and datetime fields, no checking is performed until the user leaves the field.

2. The user then leaves the field by tabbing out of it, placing the cursor on another field, or saving the page. For a date or date time field, the Clintrial software ensures at this point that the value can be interpreted as a date or datetime.
3. If the item is a verbatim text item of a coding target, then the associated coding-related fields are cleared.
4. If the item has an associated Value Changed procedure, then the procedure is run. If the procedure returns a new value, then the Clintrial software checks the item's data type. For numeric values, the new value is verified against the item's database format.

   ***Note*:** The procedure does not run if the user has not entered a value in the field. Also, the procedure does not run when a value changes as a result of another data-entry processing procedure.
5. The following checks are performed in the order specified below. If there is no Value Changed procedure attached to the item, or the Value Changed procedure o_result parameter equals 0, then the checks are performed on the value that was entered in the field. If there is a Value Changed procedure attached to the item and its o_result parameter equals 1 or 2, then the checks are performed on the value returned by the procedure.

   a. If the item is a master key, it is checked against its original value. If the original value is not null, then the new value cannot be null.
   b. If the item has an attached codelist, the Clintrial software checks that the value has not been deleted from the code list.
   c. If the item is a master key with a non-null value, the Clintrial software checks that the master key value is unique on the study page.
   d. If the item has upper or lower bounds, the Clintrial software checks that the value is within these bounds.

Note: You do not need to grant the EXECUTE privilege to each Enter user account, as you do for each user account to be able to use functions, procedures, or packages used in validation procedures. The Clintrial software automatically grants user accounts the EXECUTE privilege for conversion procedures.

Format

The format of a Value Changed procedure is as follows:

```
PROCEDURE procedure-name
  i_protocol VARCHAR2,
  i_panel VARCHAR2,
  i_table VARCHAR2,
  i_colname VARCHAR2,
  i_ct_recid VARCHAR2,
  i_colvalue VARCHAR2,
  i_itemvalues VARCHAR2,
  o_result OUT INTEGER,
  o_new_value OUT VARCHAR2,
  o_message OUT VARCHAR2,
  o_new_itemvalues OUT VARCHAR2)
 IS
 BEGIN
  PL/SQL CODE
END procedure-name;
```

Note: These are the same arguments as used for conversion procedures in prior releases of the Clintrial software. For Value Changed procedures in 4.5, you must include the o_new_itemvalues argument in the procedure call, but you do not need to use it.

Arguments

The following table lists the arguments:

Argument:	**Data type:**	**Description:**
i_protocol	VARCHAR2	Name of the protocol containing the item to which the procedure is attached.
i_panel	VARCHAR2	Name of the panel.
i_table	VARCHAR2	Indication of the table type: • UPDATE • DATA
i_colname	VARCHAR2	Name of the item.
i_ct_recid	VARCHAR2	Value of the CT_RECID system item.

Argument:	Data type:	Description:
i_colvalue	VARCHAR2	Value entered by the user (not the value of the item in the database).
i_itemvalues	VARCHAR2	List of item name and item value pairs for all items displayed in Enter for the record. To unpack this string, use the following two procedures: • CT_STRING.INIT_NAME_VALUE-_ARRAYS to create an array of values from this string. • CT_STRING.GET_ARRAY_VALUE to get the value of an item from the array created by INIT_NAME_VALUE_ARRAYS.
o_result	INTEGER	0 — No value is placed in the field and focus remains on the field. 1 — The value entered by the user is placed in the field, and focus moves to the next field in the tab sequence. 2 — The new value calculated by the procedure is placed in the field, and focus moves to the next field in the tab sequence.
o_new_value	VARCHAR2	New value to be placed in the field if the o_result is 2.
o_message	VARCHAR2	Message text, if any, to be displayed.
o_new-_itemvalues	VARCHAR2	This argument was used for conversion procedure in prior releases of the Clintrial software. For Value Changed procedures in 4.5, you must include the o_new_itemvalues argument in the procedure call, but you do not need to use it.

PL/SQL code

In a Value Changed procedure, you can use any of the procedures in the CT_EVENT package, which is described in "Event utility functions" on page 359.

Note: Changes to the database should not be committed by a Value Changed procedure.

Page-related procedures

When procedures are run

When the Enter user opens a study page, the following procedures are run automatically (in this order):

- Page Opened procedure.
- Initializing Page Section procedures, in the order of page sections represented on the study page.

When the Enter user saves data on a study page (regardless of whether the user closes the page), the following procedures are run automatically (in this order):

- Saving Page Section procedures, in the order of page sections represented on the study page.
- Page Saved.

Note: Procedures related to page sections are run when a study page is opened or saved, not when the user navigates to or from the particular page section. Initializing Page Section procedures are not run for page sections that are read-only.

Format

The format of a procedure attached to a page or page section is as follows:

```
PROCEDURE procedure-name
i_protocol VARCHAR2,
i_layout_name VARCHAR2,
i_pane_usage_seq VARCHAR2,
i_table VARCHAR2,
i_page_status VARCHAR2,
i_keys VARCHAR2,
o_result OUT INTEGER,
o_message OUT VARCHAR2)
IS
 BEGIN
  PL/SQL CODE
END procedure-name;
```

Arguments

The following table lists the arguments passed into and out of the page-related data-entry processing procedure using the format described in the previous section:

Argument:	Data type:	Description:
i_protocol	VARCHAR2	Name of the protocol containing the item to which the procedure is attached.
i_layout-_name	VARCHAR2	Name of the page template.
i_pane-_usage_seq	VARCHAR2	Number indicating the page section in the page template. This is the value of the PANE_USAGE_SEQ item in the CTSDD.PAGE_USAGE table. For procedures related to a page template, this argument is 0.

Argument:	Data type:	Description:
i_keys	VARCHAR	List of key item name and item value pairs that includes the subject item, block key item, page key item, and, if defined, the block repeat key item, page repeat key item, and subset page section key item. (The list does not include the master key item or detail key item.) To unpack this string, use the following two procedures: • CT_STRING.INIT_NAME_VALUE_ARRAYS to create an array of values from this string. • CT_STRING.GET_ARRAY_VALUE to get the value of an item from the array created by INIT_NAME_VALUE-_ARRAYS.
i_page-_status	VARCHAR2	0 — The page is new. 1 — The page already exists. 2 — Verify mode; the page already exists. -1 — Test mode in Design.
i_table	VARCHAR2	Indication of the table type: • UPDATE • DATA
o_result	INTEGER	For Page Opened events: 0 — Stop processing and close the page. 1 — Continue processing. For Initializing Page Section and Saving Page Section events: 0 — Stop processing. 1 — Continue processing. For Saved Page and Page Deleted events: 1 — Continue processing.
o_message	VARCHAR2	Message text, if any, to be displayed.

PL/SQL code

In an Initializing Page Section procedure, you can use any of the procedures in the CT_EVENT package (described in "Event utility functions" on page 359) *except* the ITEM_FOCUS, ITEM_FOCUS_RPT, and SECTION FOCUS procedure.

Note: Changes to the database should not be committed by an Initializing Page Section.

The CT_EVENT package is *not* available to the Page Opened, Page Saved, Page Deleted, and Saving Page Section events.

Note: Only panel items can be included. Inclusion of context items can cause the procedure to fail.

Attaching data-entry processing procedures

Attaching a procedure to a page template

To attach a data-entry processing procedure to a page template:

1. From Design's **Objects** menu, select **Page Template**.
2. From the **Page Template** menu, select **Modify**. Then, from the **Page Template** menu, select **Page Template Events**. The Page Template Events dialog box opens.
3. In the Procedure Name field for the appropriate event (Page Opened, Page Saved, or Page Deleted), enter the name of the data-entry processing procedure.

   If the procedure is in a package, enter:

   owner-account.package-name.procedure-name

   or:

   synonym-name.procedure-name (if there is a public synonym for the package)

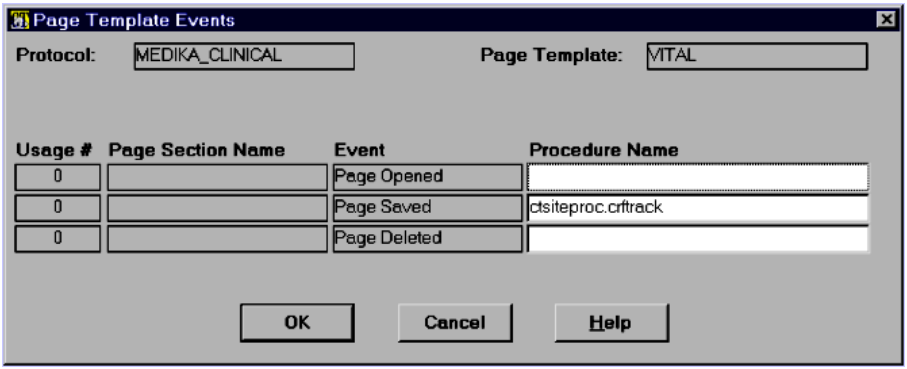
   If the procedure is *not* in a package, enter:

   owner-account.procedure-name

   or:

   synonym-name (if there is a public synonym for the procedure)

In the following example, there is a crftrack procedure attached to the VITAL page template. There are no page sections listed in the dialog box because you are working with the page template.



4. Test the procedure in page template test mode, or in Enter.

Attaching a procedure to a page section

To attach a data-entry processing procedure to a page section:

1. From the **Objects** menu, select **Page Template.**
2. From the **Page Template** menu, select **Modify**.
3. Select a page section in the page template. Then, from the **Page Template** menu, select **Modify Page Section Events**. The Page Template Events dialog box opens.
4. In the Procedure Name field for the appropriate event (Initializing Page Section or Saving Page Section), enter the name of the data-entry processing procedure.

   If the procedure is in a package, enter:

   owner-account.package-name.procedure-name

   or:

   synonym-name.procedure-name (if there is a public synonym for the package)

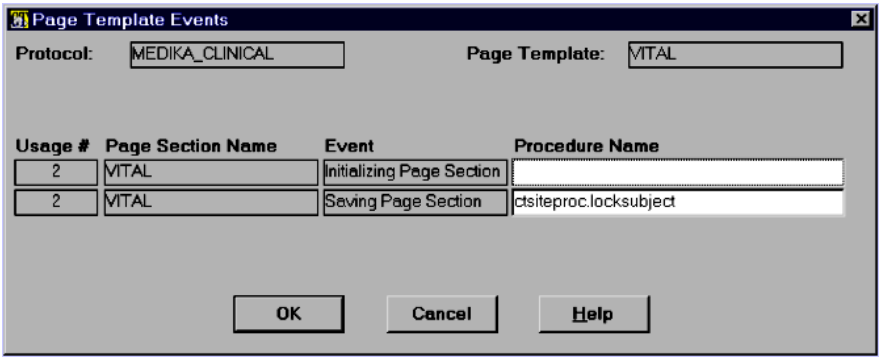
   If the procedure is *not* in a package, enter:

   owner-account.procedure-name

   or:

   synonym-name (if there is a public synonym for the procedure)

In the following example, there is a locksubject procedure attached to the VITAL page template:



5. Test the procedure in page template test mode, or in Enter.

Attaching a procedure to an item

To attach a data-entry processing procedure to an item on a study page:

1. From Design's **Objects** menu, select **Page Section**.
2. From the **Page Section** menu, select **Modify**.
3. Place the cursor in the field to which you want to attach the procedure, and, from the **Design** menu, select **Attributes**.
4. In the Value Changed Procedure field for the appropriate item, enter the name of the procedure.

   If the procedure is in a package, enter:

   owner-account.package-name.procedure-name

   or:

   synonym-name.procedure-name (if there is a public synonym for the package)

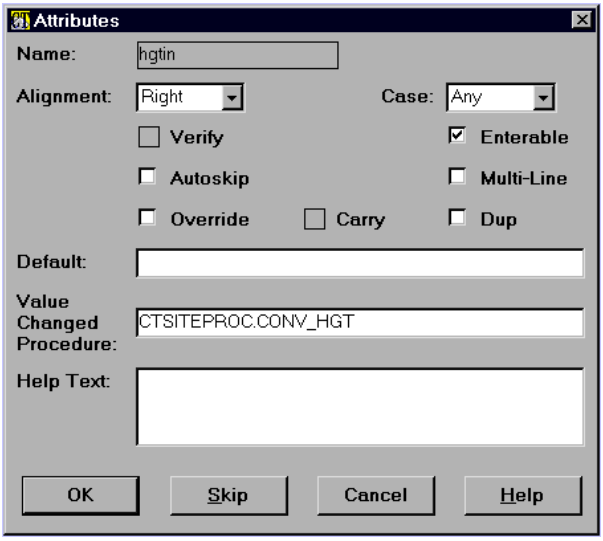
   If the procedure is *not* in a package, enter:

   owner-account.procedure-name

   or:

   synonym-name (if there is a public synonym for the procedure)

In the following example, there is a conv_hgt procedure attached to the HGTIN item:



5. Test the procedure in page template test mode, or in Enter.

Examples

Example 1: itemfocus procedure

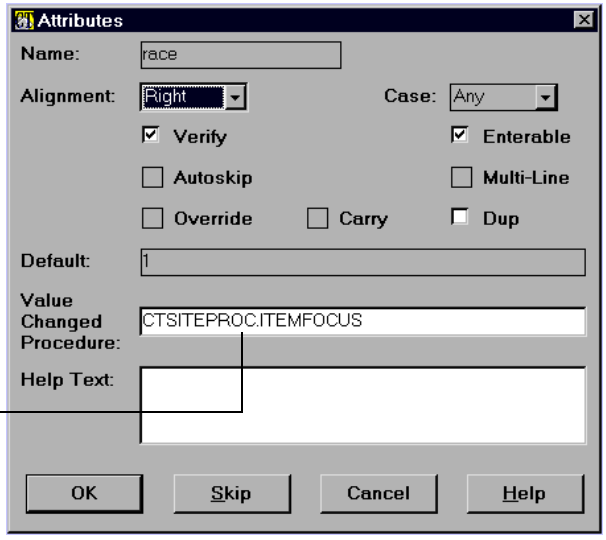
Suppose that the DMG panel includes the RACE, RACEOTH, and ALLERG items, and the page section includes those items in that order. You want a site-specific procedure to move the cursor to the RACEOTH item if '5' is entered for RACE. Otherwise, you want the cursor to skip the RACEOTH field and go to the next item in the page section, which is ALLERG.

You could create the following procedure:

```
CREATE OR REPLACE PROCEDURE itemfocus
  i_protocol VARCHAR2,
  i_panel VARCHAR2,
  i_table VARCHAR2,
  i_colname VARCHAR2,
  i_ct_recid VARCHAR2,
  i_colvalue VARCHAR2,
  i_itemvalues VARCHAR2,
  o_result OUT INTEGER,
  o_new_value OUT VARCHAR2,
  o_message OUT VARCHAR2,
  o_new_itemvalues OUT VARCHAR2)
IS

BEGIN

if i_colvalue = '5'
then

ct_event.item_focus('RACEOTH');
else
ct_event.item_focus('ALLERG');
end if:
o_result := 1;

END itemfocus;
/
```

You would then do the following:

1. Compile the procedure in the Oracle account that stores site-specific or protocol-specific procedures. This example assumes that you compile the procedure in a site-specific account named CTSITEPROC.
2. In Design, attach the itemfocus procedure to the RACE item in the DMG panel:



Or, enter the name of the public synonym, if one exists.

Example 2: convertweight procedure

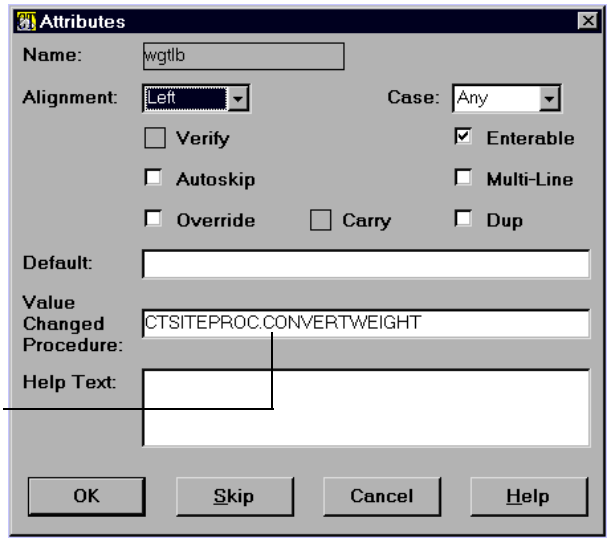
Suppose that the VITAL panel contains the items WGTLB and WGTKG. In the Vital Signs page section, you want a site-specific procedure to convert the value entered for the WGTLB item to kilograms and place the result in the WGTKG item.

You could create the following procedure:

```
CREATE OR REPLACE PROCEDURE convertweight
  i_protocol VARCHAR2,
  i_panel VARCHAR2,
  i_table VARCHAR2,
  i_colname VARCHAR2,
  i_ct_recid VARCHAR2,
  i_colvalue VARCHAR2,
  i_itemvalues VARCHAR2,
  o_result OUT INTEGER,
  o_new_value OUT VARCHAR2,
  o_message OUT VARCHAR2,
  o_new_itemvalues OUT VARCHAR2)

IS

x number(4);
myValueList ct_string.NameValueList := ct_string.NameValueList();


BEGIN

x := to_number(i_colvalue);

ct_string.add_element(myValueList,'WGTKG',to_char((x  * .454));
ct_string.set_item(myValueList);

o_result := 1;

END convertweight;
/
```

You would then do the following:

1. Compile the procedure in the Oracle account that stores site-specific or protocol-specific procedures. This example assumes that you compile the procedure in a site-specific account named CTSITEPROC.
2. In Design, attach the convertweight procedure to the WGTLB item in the VITAL panel:



Or, enter the name of the public synonym, if one exists.

Part III: Common Information

18 Data Format

Data types

What is a data type?

A *data type* is an item attribute that indicates the format in which a value for an item is entered and displayed in the Clintrial software.

Valid data types

The following table lists valid data types:

Data type:	Description:	Example:
FIXED	Integer, which optionally begins with a plus or minus sign.	100
FLOAT	Number, which optionally includes a decimal point and optionally begins with a plus or minus sign.	98.6
TEXT	String of characters, including letters, numbers, punctuation, spaces, or special characters.	Center 101
DATE	Combination of day, month, and year.	05/01/2000
DATETIME	Date and time, separated by a blank space.	05/01/2000 21:20:00

The format in which dates are entered and displayed is determined by your PC's date setting. To ensure Y2K compliance, set the date setting (a regional setting) on your PC to a four-digit year format, and enter four digits years.

If your PC's date setting is set to a two-digit year, the Clintrial software interprets the entered year as follows:

- 00 through 49 — The 2000s
- 50 through 99 — The 1900s

Database formats

What is a database format?

A *database format* is an item attribute that determines the format in which a value for an item is stored in the Clintrial software database, that is, Oracle database tables.

Valid database formats

The following table lists valid database formats:

Database format:	Description:
VARCHAR2(*n*)	Character string of up to 2,000 characters, where *n* indicates the number of characters.
DATE	A date or datetime, in Oracle's own internal format. Years are stored as four-digit years.
NUMBER(*xx*)	Any number with a maximum of *xx* digits with no decimal places. The maximum number of digits is 10. For example, if *xx*=5, the number could be 12345 or -12345.
NUMBER(*xx*,*yy*)	Any number with a maximum of *xx* digits, where *yy* indicates the number of decimal places, that is, digits to the right of the decimal point. The maximum number of digits is 10. For example, if *xx*=5 and *yy*=2, the number could be 123.45 or -123.45.

19 Naming Clintrial Software Objects

Naming conventions

When naming a Clintrial software object, use the following conventions:

- Do not exceed 20 characters.
- Ensure that the first character is alphabetic.
- Use either alphabetic or numeric characters, or both.
- Do not use any special characters, except the underscore (_).
- Do not use reserved words.

Object names are not case-sensitive.

Reserved words

Clintrial software reserved words include the following:

- The names of system items (MERGE_DATETIME, STATUS, ENTRY_ID, ENTRY_DATETIME, CT_RECID, DB_ID, CT$REASON and SUBJECT_ID).
- The words TODAY, EMPTY and the letters CT.
- Words designated as reserved in PL/SQL.
- Words designated as reserved in SQL.
- Words designated as reserved in Oracle 10g.

20 Restricting Records

Restricting records based on a SQL restriction

For some Clintrial software tasks in which you work with records, you can select records based on one of the following:

- A SQL restriction
- Flags and notes
- Date and time

What is a SQL restriction?

A *SQL restriction* is a SQL WHERE clause that is used to restrict the records with which you are working to those that meet specified conditions. The SQL restriction can consist of one or more valid SQL conditions connected by logical operators.

For example, suppose that you want to work with only records for which the value of the item SEX is F, and the value of the item AGE is less than 18 or greater than 65. You can specify the following SQL restriction, which consists of two conditions connected by the logical operator AND:

SEX = 'F' AND (AGE < 18 OR AGE > 65)

The first condition, SEX = 'F', consists of one expression. The second condition, AGE < 18 OR AGE > 65, consists of two expressions.

There is, however, a limitation to the kind of cross-panel sub-query you can add to a SQL Restriction in Clintrial. A sub-query which needs to use an alias for the primary panel (the panel selected for the batch job, as shown in the SELECT clause below), cannot be implemented. This is due to a limitation in Clintrial.

Note: When you specify an SQL Restriction for a batch job, the SQL statement is generated by appending the restriction to the system-generated SELECT clause:

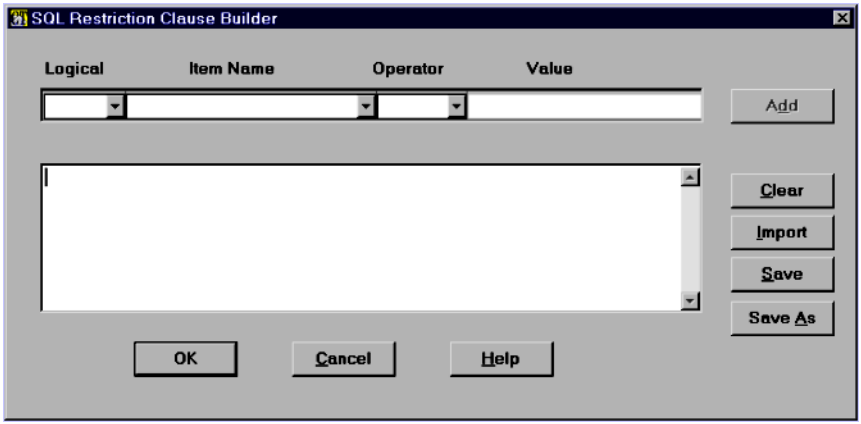
SELECT * FROM PROTOCOL.PANEL_UPDATE WHERE;

The content of the SELECT clause is always as shown above, no matter what kind of SQL restriction is used.

Caution: When sub-queries are used in SQL Restriction inappropriately, the batch job will run, and the user will interpret the batch log as saying that the restriction worked, when it did not.

How to create a SQL restriction

You can create a SQL restriction using the SQL Restriction Clause Builder dialog box:



You can also import a SQL restriction that you previously saved.

For information on how to use the SQL Restriction Clause Builder dialog box, see the Help.

Comparison operators

The following table lists the comparison operators that you can use within a SQL expression:

Comparison operator:	**Description:**	**Example:**
=	Equal to	INVNO = 316
!= <> ><	Not equal to	INVNO != 316
>	Greater than	RESULT > 120
>=	Greater than or equal to	RESULT >= 120

Comparison operator:	Description:	Example:
<	Less than	RESULT < 120
<=	Less than or equal to	RESULT <= 120
BETWEEN... AND	Between or equal to specified values	AGE BETWEEN 17 AND 65
IN	In a specified list of values	VISITNUM IN (1,2,3)
LIKE	Like a specific value	DRUGNAME LIKE 'ANTI%'
IS NULL	Has no value	LABNAME IS NULL
NOT BETWEEN... AND	Not between or equal to specified values	AGE NOT BETWEEN 17 AND 65
NOT IN	Not in a specified list of values	VISITNUM NOT IN (1,2,3)
NOT LIKE	Does not have a specified pattern	DRUGNAME NOT LIKE '%ANTI%'
IS NOT NULL	Has a value	LABNAME IS NOT NULL

Logical operators

The following table lists the logical operators that you can use to connect conditions:

Logical operator:	Description:
NOT	Evaluates to TRUE if the following condition is FALSE
AND	Evaluates to TRUE if both conditions are TRUE
OR	Evaluates to TRUE if either condition is TRUE

Syntax

In a SQL restriction:

- You can use any valid SQL functions.
- You can use any valid SQL wildcard characters.
- Do not specify the SQL keyword WHERE.
- Do not end a restriction with a semicolon (;).

Levels of precedence

The levels of precedence among SQL operators are:

- Comparison operators
- The logical operator NOT
- The logical operator AND
- The logical operator OR

You can use parentheses in an expression to override the operator precedence.

Items

Within a SQL expression:

- You can compare items to either specified values or to the values of other items.
- You can refer to the system items STATUS, ENTRY_DATETIME, ENTRY_ID, MERGE_DATETIME, CT_RECID, DB_ID, SUBJECT_ID, and CTS$REASON. These items exist for all panel types. If you refer to STATUS, use the numeric code for the record status.
- You must use the item names, rather than the field names. For example, if the Gender field on a study page is associated with the item SEX, specify SEX (not Gender) in the restriction clause.
- For items of data type TEXT, you must enclose the value in *single* quotation marks (for example, SEX = 'MALE'). Also, the case of the value that you specify must match the case (upper or lower) of the value in the database.
- For items of data type DATE or DATETIME, you must enclose the value in single quotation marks.

- For items of the data type DATE or DATETIME, use the format that is specified for by the server's setting for the Oracle parameter NLS_DATE_FORMAT. You can use the SQL function TO_DATE to change the specified date to the format in which dates are stored in the database.

Restricting records based on flags and notes

Flags

When performing certain tasks in the Clintrial software, you can restrict the records with which you are working to those that have specified flags.

Note: Only records (not items or observations) with the specified flag are selected.

Notes

When performing certain tasks in the Clintrial software, you can restrict the records with which you are working to those that have specified notes.

Note: Only records (not items or observations) with the specified note are selected.

Restricting records based on date and time

What is a date and time range?

When performing certain tasks in the Clintrial software, you can restrict records on the basis of a date and time range. The date and time that you specify are compared to the value of a record's MERGE_DATETIME (a system item) for clinical data records.

For tasks that allow restriction on the basis of date and time, you enter a From date and time or a To date and time.

From date

If you specify a From date and time, records whose ENTRY-_DATETIME is on or later than the specified date and time will be included.

If you do not specify a From date, there is no lower limit on the ENTRY_DATETIME of records to be selected.

To date

If you specify a To date, records whose ENTRY_DATETIME is on or earlier than the specified date and time will be included.

If you do not specify a To date, there is no upper limit on the ENTRY_DATETIME of records selected.

21 Using SQL in the Clintrial Software

Types of SQL statements

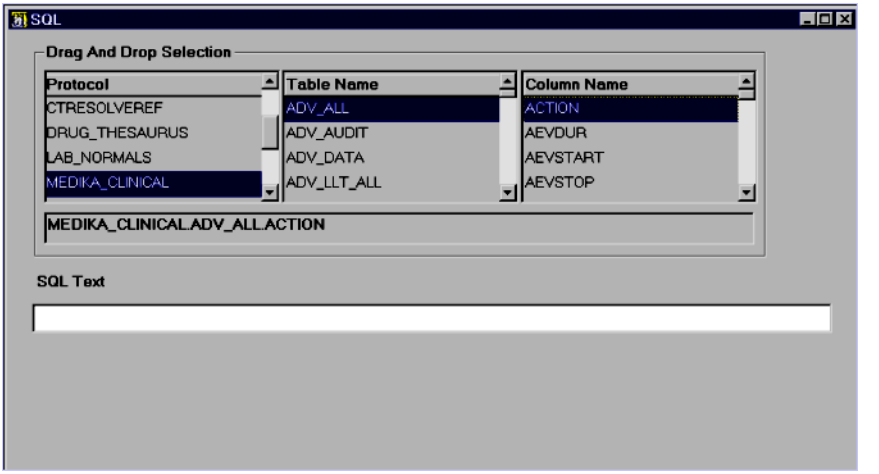
Using SQL from within the Clintrial software, you can run the following types of SQL statements:

- A SELECT statement, which retrieves from the database those records that meet specified conditions.
- A DESCRIBE statement, which displays and describes the columns in a database table or view.

See your Oracle SQL documentation for details about specifying SQL statements.

How to use the SQL command

From the **Tools** menu, select **SQL**. The SQL window opens:



For information on how to use the SQL window to specify SQL statements, see the Help.

Note: You use the **File** menu's **Import** command to import an existing SQL statement that you have saved.

Database structures

When using SQL, you need to specify full table names (including the account name) and column names. For information about accounts, tables, and columns, see Part I of this guide.

Often, you use SQL to work with protocol accounts. In this case:

- The account name is the name of the protocol.
- The table name is a panel name followed by _UPDATE, _DATA, or _AUDIT. The *panel-name*_ALL syntax includes records from both the update and the data tables
- The column name is the name of an item in the panel.

SELECT statement syntax

The syntax of a SELECT statement is:

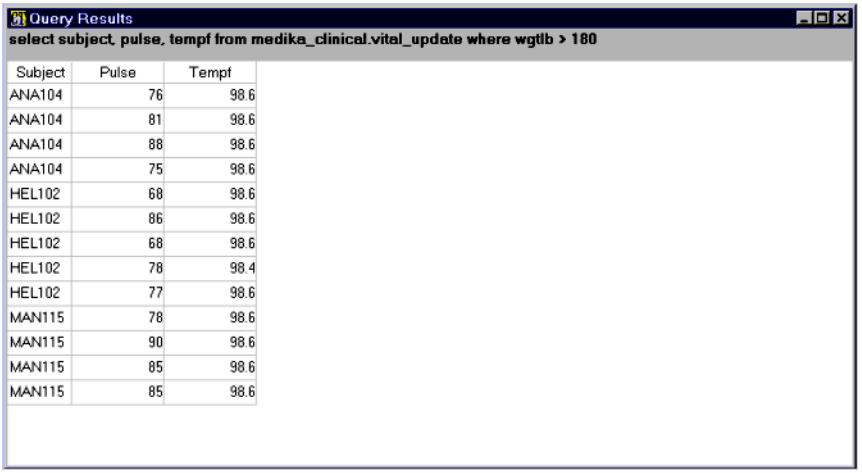
SELECT *column* FROM *account-name.table-name* WHERE *restriction-clause*;

You must end the statement with a semicolon (;). The WHERE clause is optional.

The following example retrieves values for the SUBJECT, PULSE, and TEMPF items from the update table for the VITALS panel in the MEDIKA_CLINICAL protocol, if the value of the WGTLB item is greater than 180:

```
select subject, pulse, tempf from medika_clinical.vital_update where wgtlb > 180;
```

The result of the statement is:


Query Results

select subject, pulse, tempf from medika_clinical.vital_update where wgtlb > 180

Subject	Pulse	Tempf
ANA104	76	98.6
ANA104	81	98.6
ANA104	88	98.6
ANA104	75	98.6
HEL102	68	98.6
HEL102	86	98.6
HEL102	68	98.6
HEL102	78	98.4
HEL102	77	98.6
MAN115	78	98.6
MAN115	90	98.6
MAN115	85	98.6
MAN115	85	98.6

Within a SQL expression:

- You must use the item names, rather than the field names. For example, if the Gender field on a study page is associated with the item SEX, specify SEX (not Gender) in the restriction clause.
- For items of data type TEXT, you must enclose the value in *single* quotation marks (for example, SEX = 'MALE'). Also, the case of the value that you specify must match the case (upper or lower) of the value in the database.
- For items of data type DATE or DATETIME, you must enclose the value in single quotation marks.
- For items of the data type DATE or DATETIME, use the format that is specified for by the server's setting for the Oracle parameter NLS_DATE_FORMAT. You can use the SQL function TO_DATE to change the specified date to the format in which dates are stored in the database.

DESCRIBE statement syntax

The syntax of a DESCRIBE statement is:

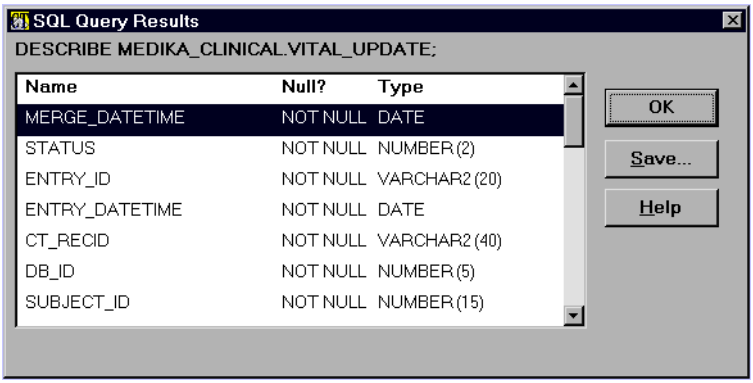
DESCRIBE *account-name.table_name*;

You must end the statement with a semicolon (;).

For example, the following statement describes the columns of the update table for the VITAL panel in the MEDIKA_CLINICAL protocol:

DESCRIBE MEDIKA_CLINICAL.VITAL_UPDATE;

The result of the statement is:



Saving statements and results

You can click **Save** to save a SQL statement or the results of a SQL statement to a file, in a variety of file formats. For information, see the Help.

22 Using Custom Menus

Overview

On the **Custom** menu, you can place commands that are shortcuts to other Windows applications (for example, Notepad or Excel) and document files (for example, text files or spreadsheets). You can then use those commands to open applications and documents directly from the Clintrial software.

To add a command, you use the **Custom** menu's **Edit Custom Menu** command. Initially, this is the only command on the Custom menu.

The commands that you add appear as additional entries on the **Custom** menu. You can specify whether the command will appear in all Clintrial software modules, or in only the module in which you create it. You can create up to eight commands of each type.

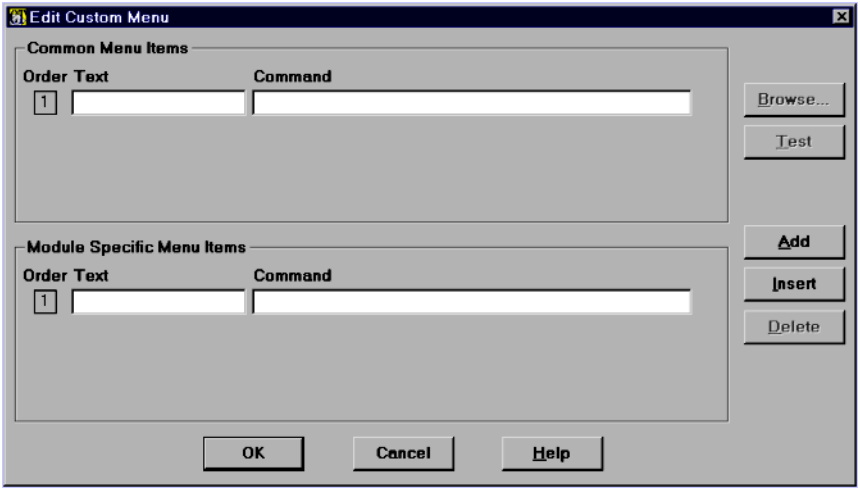
The Custom menu entries are stored in your computer's Windows Registry. Thus, they appear only on the computer on which they are created. If you are running Windows NT, the **Custom** menu entries are also specific to the user who created them.

Defining the Custom menu

To add an entry to the **Custom** menu, or to modify existing entries:

1. From the **Custom** menu, select **Edit Custom Menu**.

   The Edit Custom Menu dialog box opens:



   In the Common Menu Items or Module Specific Menu Items section of the dialog box, use the **Add**, **Insert**, and **Delete** buttons to create or remove entries.

2. In the Text field, enter the name of the command to be added to the **Custom** menu. To create an accelerator key, preface one of the letters in the name with an ampersand (&). For example:

   &Notepad

3. In the Command field, enter the path and name of the application that the shortcut will run. If the shortcut is to a document, the Command field must also include the path and file name:

   For example:

   c:\windows\notepad.exe

   or:

   c:\windows\notepad.exe d:\enter\notes.doc

   Note: If a directory name in the path contains a space, the path must be placed in quotes. For example:

   “c:\Program Files\winedit\winedit.exe”

4. To test whether a command works as it should, select it and click **Test**.
5. When you are satisfied with the **Command** menu, click **OK**.

Replacement variables

You can use the following replacement variables in the command that you specify in the Edit Custom Menu dialog box:

- CTS$USERNAME — Substitutes the user name of the current user.
- CTS$DATABASE— Substitutes @ followed by the value that you entered in the Database field in the Database Connection dialog box when you log in to the Clintrial software.
- CTS$PROTOCOL — Substitutes the name of the current protocol.

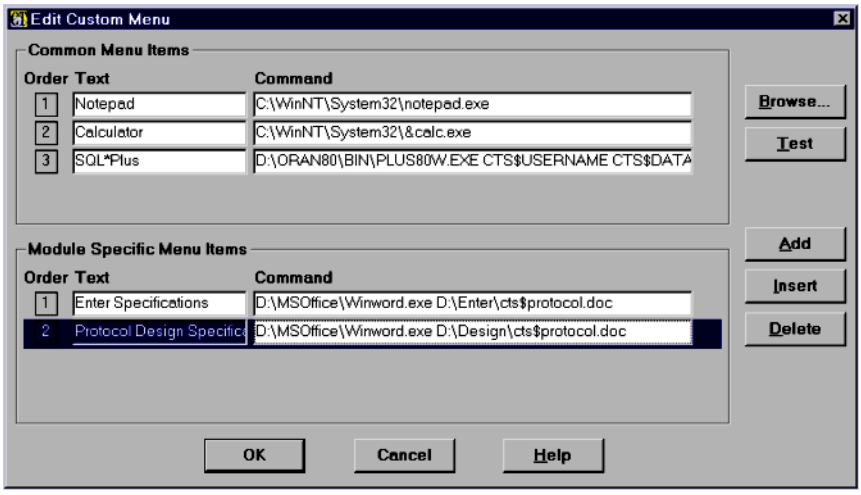
The following example logs you in to SQL*Plus as the current user in the current database instance:

```
c:\orant80\bin\plus80w.exe CTS$USERNAMECTS$DATABASE
```

Example

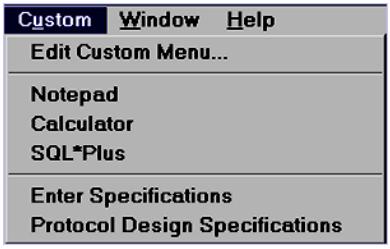
In the following example from Design, the Notepad and Calculator applications have been added to the **Custom** menu for all the Clintrial software modules. For Design only, the **Custom** menu includes the Enter Specifications and Protocol Design Specifications commands, which open document files using Word.

Note: This example assumes that there is a Word document named *protocol-name*.doc at d:\enter and another document with the same name at d:\design.

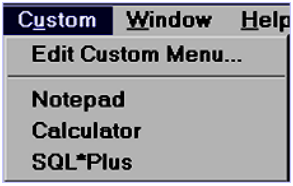


The resulting **Custom** menu appears as follows:

In Design



In other modules



23 Running Batch Jobs

Submitting a batch job

What is a batch job?

In Manage, you can perform many tasks either immediately or as a batch job. When you submit a batch job, you can specify a time at which the job should run, or intervals at which the job should run.

How to submit a batch job

To perform a task as a batch job, check Submit Batch in the active window and specify values for the following fields:

- Submit At — The date and time at which you want the batch job to begin
- Submit Every — Optionally, the time interval at which the batch job should run after the initial job (that is, the job that begins when specified in the Submit At field); if null, the batch job runs only once.

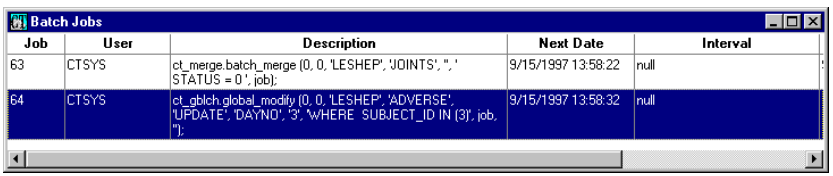
Using the batch job queue

What is the batch job queue?

The batch job queue shows currently submitted Manage batch jobs.

How to display the batch job queue

From the **Tools** menu, select **Batch Job Queue**. The Batch Jobs window opens:



You can use the **View** menu to filter or sort the batch job queue.

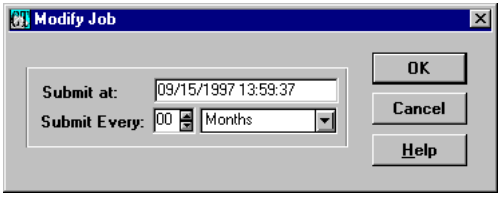
Batch job queue contents

The Batch Jobs report provides the following information:

- Job — Job number
- User — User name
- Description — SQL restriction for the job
- Next Date — Next date and time the batch job is scheduled to run
- Interval — Space of time that indicates whether batch job will run on an hourly, daily, weekly, or monthly basis
- This Date — Current date
- Last Date — Date and time the last batch job ran

How to modify entries

Select the entry you want to edit, and from the **Edit** menu, select **Modify**. The Modify Job dialog box opens:



Specify values for the following fields:

- Submit At — The date and time at which you want the batch job to begin.
- Submit Every — Optionally, the time interval at which the batch job should run after the initial job (that is, the job that begins when specified in the Submit At field); if null, the batch job runs only once.

How to run a batch job immediately

Select the batch job that you want to run, and from the **File** menu, select **Run**. The batch job runs immediately if there are no other jobs already running.

To find out if jobs are running, execute the following SELECT statement:

SELECT job_id, job_type, protocol FROM cts.job_log WHERE overall_status='RUNNING';

How to delete entries

Highlight the entry you want to delete from the batch job queue, and from the **Edit** menu, select **Delete**.

The above is all that is necessary to delete a batch job that has not yet started running. If it has already started running, you first must delete it as above, but then you must switch over to Oracle and kill the session there. For details on how to kill a session in Oracle, see your Oracle documentation.

Note: A session running unattended may cause data corruption. Do not allow running batch jobs deleted from the queue to continue to run in Oracle.

24 Glossary

In this glossary you will find definitions related to the following Clintrial software modules:

- Admin
- Design
- Enter
- Manage
- Retrieve
- Classify
- Lab Loader
- Resolve

There is a separate glossary for Multisite.

Note: You can also access an online version of this glossary using the Clintrial software Help.

A

access level

A level of security that determines what type of access the user has to the activities defined by an access right. For example, the Full access level allows complete access to the activities described in an access right. *See also* access right.

access right

A predefined set of Clintrial software activities that can be associated with a usergroup or a user. Some access rights relate to activities that require access to protocols and must be associated with a protocol as well as with a usergroup or user. Access rights in each Clintrial software module determine which combinations of activities users can perform. There are two types of access rights: non-protocol access rights and protocol access rights. *See also* access level, non-protocol access right, protocol access right.

account

See system account, protocol account, user account.

Ad Hoc Query

A query method in Retrieve that allows you to use a graphical query builder to create lists of subject records, then browse those subject records, or save detail data for those subjects.

aggregated codelist

A codelist that is stored in a single Oracle database table with other codelists. *See also* codelist, unaggregated codelist, view codelist.

attribute

A characteristic of an object that defines the object. The definition of an object is the set of its attributes. For example, the attributes of an item include its name, database format, and whether it is required. *See also* object.

auditing

The tracking of changes to clinical data, and of notes associated with clinical data. Depending on the audit start point, auditing tracks changes to records and notes in the update table, the data table, or both. *See also* audit start point.

In revision control, the tracking of changes to metadata in the source object or the destination object. *See also* source object, destination object.

audit start point

The data management activity after which auditing begins, set by the designer. For example, auditing can begin after data-entry, verification, validation, validity, or merging. *See also* auditing.

audit table

An Oracle database table that stores copies of clinical data records as they existed before modification or deletion. There is one audit table for each installed panel. The name of the audit table is *panel-name*_AUDIT; for example, LAB_AUDIT. *See also* data table, update table.

automatic coding

A method of coding in which the Clintrial software uses an algorithm to search the coding thesaurus and assign an appropriate code to verbatim text. *See also* thesaurus algorithm, coding thesaurus, interactive coding, verbatim text.

automatic skipping (Autoskip)

An optional feature that advances the cursor automatically from one field to the next once the maximum number of characters has been entered. Autoskip only affects fields for which the designer has set the Autoskip attribute. Typically, these are fields for which there is a fixed entry length (for example, date fields).

B

base protocol

The clinical data protocol on which a view protocol is based. *See also* clinical data protocol, view protocol.

batch ID

An identifying value assigned to an Oracle batch job. *See also* batch job.

batch job

A process that is submitted to an Oracle batch job queue, rather than run interactively. Batch jobs can be run immediately, or they can be run later at a specified date and time, or a regularly scheduled interval. *See also* batch ID.

batch loading

The process of taking data that is stored in an ASCII file and placing that data directly into a Clintrial software database table. *See also* control file, input file.

blind verification

A way of checking that clinical data has been entered correctly. To perform blind verification, you reenter values in fields in a study page that require verification. The Clintrial software checks the reentered data when you exit a study page, attaches a VERIFICATION/AUTOFLAG flag to any conflicting items, and adds an entry for each item to the Verification Report. Blind verification is performed as heads-down data entry. *See also* interactive verification, verification.

block

A group of related study pages in a study book. Blocks usually represent subject evaluation checkpoints, such as subject visits. For example, a block named VISIT1 can include the following study pages: Vital Signs, Concomitant Medications, and Laboratory Exams. *See also* study book, study page.

block key item

A visit-related context item that is a key for identifying data by block. For a nonrepeating block, the value of the block key item uniquely identifies the block; for a repeating block, the values of the block key item and the block repeat key item uniquely identify the block.
Typically, the block key item is the context item for identifying the visit. For example, you may use the Laboratory Exams page to collect laboratory exam data during several subject visits. The block key item distinguishes which visit's data to display in a study page. *See also* block key value, block value, block repeat key item, block repeat key value, special context item.

block key value

The value given to the block key item, which must be a visit-related context item. You must specify the block key value in Design when you create a block in the study book layout
editor. The block key value can be modified only if there is no clinical data that refers to the value, and if the block has no study pages. *See also* block key item, block repeat key item, block repeat key value, block value, special context item.

block repeat key item

A visit-related context item that is the key, with the block key item, for uniquely identifying data in a repeating block. *See also* block key item, block key value, block repeat key value, block value, special context item.

block repeat key value

The value given to the optional block repeat key item, which must be a visit-related context item. You must specify the block repeat key value in Design when you create a repeating block in the study book layout editor, or in Enter when you create a repeating block. *See also* block key item, block key value, block repeat key item, block value, special context item.

block title

A name or a phrase that describes a block as it appears in a study book.

block value

Optional initial values given to any visit-related context item other than the block key item or block repeat key item. You can specify the block values when you modify a block in the study book layout editor. *See also* block key item, block key value, block repeat key item, block repeat key value, context item.

C

carried value

In data entry, a value that is carried from one field in a previous study page to the same field in the current study page. Carried values are displayed only in context page sections of study pages. If you can place the cursor in a field that contains a carried value, you can change the initial value to a new value. *See also* context page section.

case report form (CRF)

A paper form used to record clinical data for study subjects. In the Clintrial software, CRFs are represented online by clinical data study books. *See also* study book, study page.

checklist

In data entry, a type of codelist used to view suggested entries for a field. *See also* codelist, value field type.

checkpoint date

A date specified in a view protocol. The Clintrial software creates the appearance of the clinical data stored in the base protocol's clinical data table as it existed on the checkpoint date. For example, a checkpoint view protocol might present the clinical data as it existed on June 1, 2000. *See also* view panel, view protocol.

clinical data

Data that is collected during clinical trials; for example, data about a subject collected on a case report form (CRF) such as demographic data, previous medications, or laboratory test results. *See also* flag, metadata, note.

clinical data management

The process of describing, managing, and reporting on the data generated by a clinical trial. *See also* clinical trial.

clinical data protocol

A protocol that can contain both clinical data and metadata definitions of Clintrial software objects. Some clinical data protocols contain only metadata definitions of Clintrial software objects. Optionally, a clinical data protocol creates clinical data tables that store clinical data. *See also* coding thesaurus protocol, dictionary protocol, Lab Loader protocol, view protocol.

clinical data table

An Oracle database table that contains clinical data or information about clinical data for an associated panel. When a panel is installed, three types of clinical data tables are created: update table, data table, and audit table.

clinical protocol

A detailed plan for a clinical study that describes how investigators will conduct each study in a clinical trial. The clinical protocol sets the guidelines for the study, describes the conditions of the study, and contains a set of forms on which clinical data will be collected. *See also* clinical study, Clintrial software protocol, protocol account.

clinical study

In clinical trials, a research procedure defined by a clinical protocol to determine the safety or efficacy of a drug, medical treatment, or medical device. *See also* clinical protocol, Clintrial software protocol, protocol account.

clinical trial

See clinical study.

Clintrial software administrator

The Clintrial software user who maintains user accounts and manages access to all Clintrial software modules using Admin. The Clintrial software administrator also sets systemwide parameters for the Clintrial software.

Clintrial software protocol

A logical container that organizes the Clintrial software objects and clinical data for a clinical study. *See also* clinical protocol, protocol account.

code

A unique alphabetic or numeric identifier for a clinical term. Codes and their descriptions are stored in coding thesaurus protocols. Codes are used to standardize the data collected from different investigators, sites, and languages to allow accurate comparison and analysis.

code field type

A field type that defines whether the code from a codelist is stored as text or as a number in the coded item in the panel. The code field type must be of the same data type as the coded item in the panel. *See also* codelist, field, value field type.

codelist

A set of codes and corresponding set of values. If a codelist is attached to an item, you can only enter codes (or values) from the codelist for the item. *See also* code field type, checklist, value field type, view codelist, subset codelist.

codelist status

A codelist attribute that can have the values Valid or Invalid. Valid indicates that the codelist can be used by Enter. Invalid indicates that the codelist cannot be used. You can specify an Invalid status for a codelist provided that it is not associated with an item and it is not the base codelist for a subset.

coding

The process of assigning a standard code from a coding thesaurus to a value that has been entered for an item. For example, an entered drug could be assigned a standard code from the WHODRL thesaurus. You can perform coding using two methods: automatic coding or interactive coding. *See also* automatic coding, coding thesaurus, interactive coding, verbatim text.

coding target

A Clintrial software object that identifies a set of items in a panel of a clinical data protocol; these items are used with a coding thesaurus protocol during coding. *See also* coding, coding thesaurus protocol, verbatim text.

coding thesaurus

A dictionary thesaurus that contains standard codes for a particular type of clinical data. There are two types of thesauruses: industry-standard and user-defined. For example, the industry-standard COSTART and WHOART thesauruses contain standard codes for clinical events. You can also create a user-defined thesaurus for your study, and add synonyms to a standard coding thesaurus. *See also* coding thesaurus protocol.

coding thesaurus protocol

A protocol that stores the database tables containing a coding thesaurus. *See also* clinical data protocol, coding thesaurus, dictionary protocol, extended thesaurus protocol, protocol account, view protocol.

container object

An object that contains other objects that exist only within the container object. For example, a panel is a container object, which contains rules, derivations, items, and coding targets.

context item

An item in a context panel that provides context information for a record in a clinical data table. Context items are included as columns in all clinical data tables defined by a panel of Type 1, 2, 3, 4, or 5. Special context items, such as the subject item, are used as keys (either alone, in combination with each other, or in combination with items in the non-context panels) to uniquely identify a record. *See also* context panel, context page section, special context item, subject item.

context page section

A section of a study page, based on the context panel, that contains context items that uniquely identify a record, such as subject and visit. Each study page in which you enter clinical data for a subject contains a context page section. *See also* context item, context panel.

context panel

A special panel that contains context items that are included as columns in all clinical data tables defined by a panel of Type 1, 2, 3, 4, or 5. There is one context panel for each protocol. The context panel must be named CONTEXT. The context panel requires a corresponding context page section that

appears once on each study page, if that study page contains page sections based on panel Types 1, 2, 3, 4, or 5. *See also* context item, context page section, panel.

control file

A file that provides instructions for how data from an ASCII input file will be batch loaded into the database table. *See also* batch loading, input file.

CT_MEDDRA protocol

A thesaurus protocol provided by the Clintrial software that provides Clintrial software support for a single-language coding using the MedDRA dictionary. You can enhance CT_MEDDRA to meet your coding needs, for example coding in multiple languages. *See also* Medical Dictionary for Regulatory Activities.

CTL_REFERENCE protocol

A thesaurus protocol set up to contain lab normal ranges and SI unit conversions.

CTPROC account

A system account that stores internal Clintrial software procedures.

CTRESOLVEREF

A reference protocol supplied during Resolve installation. It contains panels, codelists, and other items that structure your use of Resolve.

CTS account

A system account that stores information about protocols, user accounts, user access rights, parameters, flag and note definitions, and other metadata that is not protocol-specific.

CTSCODES account

A system account that stores codelist information.

CTSDD account

A system account that stores protocol-specific data dictionary information such as item and panel definitions.

CTS$LOAD_protocol-name account

A system account that handles privileges for batch loading. There is one CTS$LOAD account for each protocol, named CTS$LOAD*_protocol-name.*

CTSYS account

A user account supplied by the Clintrial software for system administration.

Custom menu

The Custom menu is a user-modifiable menu on which you can place shortcuts to other Windows applications (for example, Notepad, Excel) and document files (for example, text files, spreadsheets). You can then use those shortcuts to open applications and documents directly from the Clintrial software.

CXFR_RECV

A system account that handles the importing of the codelists from the codelist.dmp file.

CXFR_SEND

A system account that handles the exporting of the codelists to the codelist.dmp file.

D

database format

The format in which a value for an item entered in the Clintrial software is stored in the Oracle database. The database format is an item attribute. The Oracle database formats are VARCHAR2(*n*), DATE, NUMBER(*xx*), NUMBER(*xx,yy*), NUMBER(*xx,0*). *See also*
attribute, data type.

database ID

Registration number of the database instance. The default is 1.

database profile

A set of limits on Oracle database resources, associated with each user account. *See also* user account.

database table

An Oracle database table that stores clinical data or metadata. *See also* clinical data, clinical data table, metadata.

data dictionary number

A number that the Clintrial software assigns to each item in an installed panel.

data dictionary table

An Oracle database table that contains metadata (objects and parameters). Data dictionary tables do not contain clinical data.

data discrepancy form

A formatted report used to present one or more queries to the clinical investigator and to capture resolution information.

data entry

The process of entering or editing clinical data using a study book.

data-entry operator

A Clintrial software user who enters data from a paper case report form (CRF) into a Clintrial software study book. The data-entry operator usually has access rights only to data in the update table. *See also* case report form (CRF).

data-entry processing procedure

A PL/SQL procedure that runs when the Enter user performs a particular action related to a study page, page section, or field. For example, a data-entry processing procedure attached to a field could convert an entered value to another value, or move the cursor to a particular field if a specific value is entered in the current field.

data manager

The Clintrial software user who manages the clinical data for a study, using Manage to perform such tasks as validating and merging records, making global modifications, or coding data.

data ownership

The ability of a site in a replication environment for a protocol or account to modify or delete data. *See also* replication.

data processing event

A Clintrial software action to which a data-entry processing procedure can be attached.

data table

An Oracle database table that stores only data that has passed validation; that is, data that is clean. There is one data table for each installed panel. The name of the data table is *panel-name*_DATA; for example, LAB_DATA. *See also* update table, validation.

data transfer

The process of moving some or all of the records in a Lab Loader source protocol to records in a clinical data destination protocol.

data type

The format in which a value for an item can be entered in a field in the Clintrial software. The data type is a required item attribute. The Clintrial software data types are TEXT, FIXED, FLOAT, DATE, or DATETIME. *See also* database format.

date field

In data entry, a field in which you can enter a value (with the data type DATE) that is interpreted as a date.

date/time field

In data entry, a field in which you can enter a value (with the data type DATETIME) that is interpreted as a combination of the date and time.

default protocol (current protocol)

The protocol in which you are currently working in a Clintrial software module. If you want to work with Clintrial software objects or clinical data in another protocol, you must change the default protocol.

default value

A predefined value for an item, displayed in a field in data entry. If you can place the cursor in a field with a default value, you can change the default value.

derivation

A PL/SQL statement that is attached to a panel, and calculates the value of an item or a temporary variable for records during validation. Derivations are part of the validation procedure that checks clinical data before it is moved to the data table. For example, if the user has entered a birth date but no age, a derivation could calculate the age based on the visit date and the birth date. *See also* rule, validation procedure.

derived item

An item whose value is calculated by a derivation when the record is validated. *See also* derivation, validation procedure.

derived value

A value that is calculated automatically and written to a derived item when a record is validated. In data entry, derived values are not displayed until a record is validated. Some fields display derived values only; you cannot enter a value in those fields. *See also* derivation, derived item, validation procedure.

designer

The Clintrial software user who designs Clintrial software studies and sets up and maintains Clintrial software objects using Design.

destination object

In revision control, an object that has been copied from another object with a connection. *See also* revision control, source object.

destination protocol

A clinical data protocol that contains the clinical data for a study. You transfer the lab data into this protocol after preparing it in the source protocol. *See also* source protocol.

detail key item

An item in the detail record that is the key for uniquely identifying the master record associated with the detail record. Users do not enter the value for the detail key item; the value is propagated from master key item in the associated master record. *See also* detail page section, detail record, and master-detail relationship.

detail page section

A page section based on the panel containing the detail key item. A detail page section may allow repeating items. The detail page section is placed on a page template with the associated master page section. In a study page, when a record is selected in the master page section, the associated detail records are displayed in the detail page section. *See also* detail key item, detail record, and master-detail relationship.

detail record

A record in the detail page section that is associated with a master record and distinguished by the value of the detail key item. The detail record may be in a page section with repeating items, in which case multiple instances of a detail record can be associated with a single master record. *See also* detail key item, detail page section, and master-detail relationship.

dictionary protocol

A protocol that contains only metadata objects created within that protocol. A dictionary protocol cannot create clinical data tables and cannot store clinical data. *See also* clinical data protocol, coding thesaurus protocol, Lab Loader protocol, parent protocol, view protocol.

discrepancy

A potential or actual data problem, such as inconsistent or missing data, identified through automated checking or manual inspection.

discrepancy flag

A flag in a flag category whose name begins DISCREPANCY_P. Using these flag categories helps automate creation of discrepancy records.

discrepancy record

A record in Resolve that describes an error found in clinical data. Discrepancy records help you track a data problem and manage its resolution. Each discrepancy record relates to a primary, or source, data record item, but may also include references to other items in the same or other data records.

discrepancy status

Assigned to discrepancy records to indicate their progress through the Resolve workflow. Discrepancy statuses allow transitions to one or more other statuses, or are terminal and close the investigation for a data error.

discrepancy transition

Regulates the order in which discrepancy statuses can be assigned to discrepancy records. By specifying the set of allowable progressions among Resolve's statuses, transitions prohibit all other status changes.

distribution

In Multisite, the movement and management of metadata objects among multiple sites. Using Multisite, you copy, or distribute, metadata objects from one site to one or more other sites, as well as control and track revisions of those objects across sites.

double entry

A way to check whether data has been entered correctly in study pages by reentering the data; also called interactive verification. *See also* interactive verification.

duplicate record

A record for which a record already exists for the same subject or visit and for the same panel.

E

enroll

To add a subject to a study. A subject must be enrolled before clinical data can be entered for the subject. *See also* enrollment panel, subject item.

enrollment panel

A panel that stores enrollment data that uniquely identifies each subject in a study. An enrollment panel contains all context items (including one item that is designated as the subject item), and optionally, other subject-related items defined for the panel. Each clinical data protocol must have an enrollment panel; the enrollment panel is required to enter clinical data interactively using study pages. *See also* enroll.

export

The process of using the Oracle Export Utility to store protocol and codelist metadata and data in files that can be imported to the same or another database instance to create copies of protocols and codelists.

extended thesaurus protocol

A Classify-specific coding thesaurus protocol designated by the CTG_THES_TYPE protocol parameter. An extended thesaurus protocol allows Classify to display additional data contained in and specific to the extended thesaurus protocol. *See also* coding thesaurus protocol, GCT_MEDDRA protocol, GCT_WHODD protocol.

F

field

The data-entry area in which values for an item can be entered on a study page. A field name may differ from the name of the actual item in the database. For example, a value for the item BIRTH_DATE might be entered in a field named Date of birth. *See also* study page.

flag

An attachment to clinical data used to label and monitor data quality problems. For example, you might attach a flag to a number that is illegible, missing, or out of the expected range. In data-entry, you can attach a flag to an observation, a record, or an item. *See also* note.

G

GCT_MEDDRA protocol

A Classify extended thesaurus protocol designed to hold MedDRA data. *See also* extended thesaurus protocol.

GCT_WHODD protocol

A Classify extended thesaurus protocol designed to hold WHO drug dictionary data. *See also* extended thesaurus protocol.

global change

The process of changing the value of one or more items in multiple records.

global deletion

The process of deleting one or more items in multiple records.

grouping item

A sorting item that also functions as a key for grouping multiple records into observations in a page section based on a Type 0 panel. *See also* sorting item.

grouping records

In screening, the process of grouping batch-loaded records into observations. *See also* batch loading, observation.

H

High Level Group Terms (HLGT)

A level of terms in the MedDRA dictionary hierarchy used to group together HLTs, and for data retrieval purposes.

High Level Terms (HLT)

A level of terms in the MedDRA dictionary hierarchy used solely for data retrieval purposes.

hot key

A keystroke combination used to issue a command to a third party imaging/workflow system to perform an action, such as display a scanned image of a Case Report Form (CRF) or Data Correction Form (DCF). Hot keys can be used while entering, editing, or viewing data on a study page in Enter or while viewing a discrepancy detail in Resolve.

I

import

The process of using the Oracle Import Utility to create copies of protocols and codelists based on protocol and codelist metadata and data in files that was exported from the same or another database instance. *See also* export, reconciliation.

input file

An ASCII file that contains data to be batch loaded into the Clintrial software. *See also* batch loading, control file.

interactive coding

A method of coding in which you can browse the coding thesaurus and assign an appropriate code to verbatim text. *See also* automatic coding, coding thesaurus.

interactive SQL

The use of SQL from within the Clintrial software to create or import a SQL statement and run it against the Clintrial software database.

interactive verification

A way to check whether data has been entered correctly in study pages by reentering the data. When you do interactive verification, the Clintrial software compares each value that you reenter on a field-by-field basis and informs you if there is a conflict between the old and the new data. Interactive verification is also referred to as double entry. *See also* blind verification,
verification.

International System of Units

A system of presenting units of measure in universally recognized formats. Also referred to as Système International (d'Unités). *See also* SI unit.

investigator

In a clinical study, the person who fills out a case report form (CRF) for a subject.

investigator note

An annotation that the investigator attaches to clinical data, consisting of a note category, note name, and note text. For example, an investigator note might be "Subject did not take full regimen of drug." *See also* investigator, sponsor note.

item

A Clintrial software object that stores a piece of data, for example, the data collected by a single field in a study page, or a single field in a batch-loaded file. Items are defined within panels, and each corresponds to one column in a clinical data table. For example, the DEMOG panel might contain the items AGE and SEX. *See also* field, panel.

L

Lab Loader protocol

A protocol for use with batch-loaded data. For more information, see the Lab Loader documentation.

LAB_NORMALS protocol

A Lab Loader source protocol set up to contain lab test results.

Lab normal ranges (also called reference values)

A set of laboratory test results that serve as reference values for subsequent laboratory test results.

local protocol

A temporary version of an imported protocol, used at the importing site to reconcile and install the imported metadata, and optionally to load imported data. The local protocol is then released to become a working protocol at the importing site. *See also* export, import, reconciliation.

lookup step

A thesaurus algorithm step that attempts to match verbatim or normalized text to a term in a thesaurus. When a single exact match is found, the code that corresponds to the term is assigned and automatic coding succeeds.

Lowest Level Terms (LLT)

The lowest hierarchical level of terms in the MedDRA dictionary.

M

mandatory item

An item for which a value is required. A value must be present in the database before a record in the update table can be merged; that is, moved to the data table. Typically, a value for a mandatory item is entered interactively, batch-loaded, or supplied by a data-entry processing procedure. *See also* screening.

master-detail relationship

A relationship between two page sections on a study page, in which each record in one page section (the master page section) can have one or more associated records in the other page section (the detail page section). During data entry, the displayed records in the detail page section are associated with the selected record in the master page section. If you have multiple records in a master page section, you can access different sets of records in the detail page section. There are two types of master-detail relationship, cross-panel and within-panel. In a cross-panel master-detail relationship, the detail page section is based on a different panel than the master page section. In a within-panel master-detail relationship, the detail page section is based on the same panel as the master page section. *See also* detail key item, detail page section, detail record, master key item, master page section, and master record.

master key item

An item in the master record that is the key for uniquely identifying the detail records associated with the master record. When a user enters a value for the master key item, that value is propagated to the associated detail records. *See also* master page section, master record, and master-detail relationship.

master page section

A page section based on the panel containing the master key item. A master page section may allow repeating items. The master page section is placed on a page template with the associated detail page section. In a study page, when a record is selected in the master page section, the associated records are displayed in the detail page section. *See also* detail key item, detail record, and master-detail relationship.

master record

A record in the master page section that is associated with one or more detail records and distinguished by the value of the master key item. The master record may be in a page section with repeating items, in which case each instance of a master record is associated with a different set of detail records. *See also* master key item, master page section, and master-detail relationship.

Medical Dictionary for Regulatory Activities (MedDRA)

A mixed-case, hierarchical thesaurus that contains terminology applicable to all phases of drug development, and to the health effects of devices.

medical reviewer

A Clintrial software user who uses query tools to retrieve data from the database for clinical review or statistical analysis.

merged data and unmerged data

Merged data is data that has been validated (cleaned) in the update table and moved to the data table. Unmerged data is data in the update table that has not yet been moved to the data table. *See also* clean data, merging.

merging

The process of moving validated data from the update table to the data table.
See also validation.

metadata

Data that defines Clintrial software objects and their relationships. *See also* clinical data.

metadata report

A report that displays the metadata associated with a particular type of database object. For example, the Items and Coding Targets Report lists item names, item attributes and coding targets for all selected panels. The set of Clintrial software metadata reports includes all the primary metadata currently in the Clintrial software data model.

N

navigation

In data entry, the process of moving through a study book to enter or edit data. *See also* navigation order, page list, subject list.

navigation order

The order in which you navigate a study book during data entry. You can set a navigation order to navigate a study book by page or by subject. The default navigation order is Navigate By Subject. The navigation order also determines the sequence in which you view subjects, blocks, and study pages in the Navigator and the way that you print study pages. *See also* navigation, page list, subject list.

Navigator

In Enter, the main browser for navigating study books and selecting subjects and study pages for data entry, editing, and printing. The Navigator displays a hierarchical listing of pages and subjects in a study book. *See also* navigation, navigation order.

non-patient data panel

A panel that stores data that is not related to a particular subject or visit, such as standard coding thesauruses, view codelists, or laboratory normal ranges. *See also* context panel, enrollment panel.

non-protocol access right

A type of access right for Clintrial software activities. A non-protocol access right is not associated with a particular protocol, but only with a usergroup or a user. For example, the System access right in Admin, which allows users to work on system parameters and security, is a non-protocol access right. *See also* access right, protocol access right.

normalcy status

A value that indicates the measure of normality of the test result as compared to the lab normal ranges.

normalized text

The result of applying one or more thesaurus algorithm transformations to verbatim text. Normalized text retains the same meaning as the original verbatim text.

note

An attachment to clinical data used to record an annotation made by an investigator or sponsor on a CRF. For example, an investigator might comment, "Subject did not take full regimen of drug." During data-entry, you can attach a note to an observation, a record, or an item. *See also* investigator note, sponsor note.

numeric field

In data entry, a field in which you must enter a number for items with the data types FIXED or FLOAT. If the data type of the item is FIXED, you cannot include a decimal point. If the data type of the item is FLOAT, you can optionally include a decimal point. For items with the data types FIXED or FLOAT, you can precede the number with a plus sign or a minus sign.

O

object

A data structure in the Clintrial software; for example, a protocol, a panel, an item, or a codelist.

object attribute

See attribute.

object browser

A window that displays a list of Clintrial software database objects of a specific type. You typically use an object browser to select the object or objects with which you want to work. Object browsers are also associated with menus that allow you to perform various operations on a selected object.

observation

A group of records in a page section that contains repeating items, or a group of records in a non-subject study book that has been configured for grouping. Any record added to the page section is automatically included in the observation. This grouping allows the records to be treated as a single set during data management tasks such as auditing and validation.

omission

A term or phrase in a verbatim text item field that fails to code when automatic coding runs. Coding fails when either no matching terms, or more than one matching term, are found for the normalized version of the verbatim text.

omission record

A record in Classify that stores information about each instance in which automatic coding fails. Used to track, review, and find solutions for values that failed to code automatically.

omission status

A status assigned to omission records to indicate your progress in finding solutions for them. Omission statuses are assigned automatically as a result of coding in Manage and actions performed in Classify.

P

page key item

A page-related context item that is the key for retrieving data in clinical data tables by the study page within a block. For a non-repeating study page, the value of the page key item uniquely identifies the page within the block; for a repeating page, the values of the page key item and the page repeat key item uniquely identify the study page within the block. Typically, the page key item is the context item for identifying the study page number within a block. *See also* page key value, page repeat key item, page repeat key value, page value, special context item.

page key value

The value assigned to the page key item, which must be a page-related context item. You must specify a page key value when you create a study page in the study book layout. The page key value can be modified only if there is no clinical data that refers to the value. *See also* page key item, page value, special context item.

page list

A list of pages that you can use to navigate a study book. For example, suppose that a study book includes study pages 1 through 4. If you know that you only want to use pages 1, 4, and 2, in that order, you can define a page list that will display those study pages in that order. *See also* navigation, navigation order.

page repeat key item

A page-related context item that is the key, with the page key item, for uniquely identifying data in a repeating page within a block. *See also* page key item, page key value, page repeat key value, page value, special context item.

page repeat key value

The value assigned to the optional page repeat key item, which must be a page-related context item. You must specify the page repeat key value in Design when you create a repeating page in the study book layout editor, or in Enter when you create a repeating page. *See also* page key item, page key value, page repeat key item, page value, special context item.

page section

A section of a study page that collects a related clinical data. The data is then stored in clinical data tables defined by a panel that is the base of the page section. For example, a study page might include a page section for demographic data (DMG panel), a page section for vital signs (VITAL panel), and a page section for physical examination result (PHYEXM panel). *See also* study page.

page template

A grouping of page sections that defines the layout of page sections in a study page. A page template consists of one or more page sections, as well as a context page section, if the other page sections are based on panel Types 1, 2, 3, 4, or 5. One or more study pages may be based on a single page template. *See also* context page section, page section.

page value

Optional initial values assigned to any page-related context item other than the page key item. You can specify the page values when you modify a study page in the study book layout editor. *See also* page key value, page value.

panel

A collection of logically related or clinically related items. For example, the EXAM panel might define items for physical examination results. When a panel is installed, three database tables are created (update, data, and audit tables) for storing clinical data. The items in the panel are the columns of the database tables. *See also* item, panel type.

panel type

A number (0 – 5) that defines how the database tables associated with the panel (the update, data, and audit tables) are structured; that is, whether there can be one or multiple records for each subject item, or subject item and block key item combination. *See also* panel.

parent protocol

A protocol that defines the default searchlist for another protocol. For a protocol that has a parent protocol, the protocol's searchlist includes its parent protocol and its parent protocol's searchlist. *See also* protocol hierarchy, searchlist.

Preferred Terms (PT)

Distinct descriptors for the MedDRA terminology categories. PTs are hierarchically located above LLTs, and are used to group together equivalent LLTs.

private query

A query in the query library that can be run only by the user who created the query. *See also* public query, query library.

programmer

Clintrial software user who uses PL/SQL to build rules, derivations, and data-entry processing procedures; integrates the Clintrial software with other applications; or uses a third-party tool to create formal reports.

protected panel

A panel with the protected attribute that cannot be accessed unless a user is granted access through Admin.

protocol

In the Clintrial software, generally used to refer to a Clintrial software protocol. *See* clinical protocol, Clintrial software protocol, protocol account.

protocol access right

A type of access right for Clintrial software activities that are specific to a protocol. A protocol access right is associated with a protocol and a usergroup or a user. For example, the Database access right in Design, which allows users to work with items and panels in a particular protocol, is associated with a protocol, and the user or usergroup. *See also* access right, non-protocol access right.

protocol account

An Oracle account that consists of database tables that hold the clinical data, flags, and notes associated with clinical data, audit and error log information, and one or more views of the clinical data stored in the protocol. *See also* clinical protocol, Clintrial software protocol, system account, user account.

protocol hierarchy

A description of the relationships among protocols that determines how protocols can share data definitions. The protocol hierarchy is structured like a data tree, defining relationships among protocols. *See also* Clintrial software protocol, parent protocol, searchlist.

protocol-name_DATATRANS

A system account that contains all the protocol clinical data when the protocol is exported.

protocol parameter

Parameters that tailor the working environment for particular protocols. Protocol parameters take their default values from corresponding system parameters. In Design, a user with access to a protocol can change that protocol's parameters. Not all system parameters have corresponding protocol parameters.

protocol type

One of five categories of protocols, used to group protocols depending on their use and properties. The protocol types are clinical data protocol, Clintrace protocol, coding thesaurus protocol, Lab Loader protocol, and view protocol. *See also* Clintrial software protocol.

public query

A query in the query library that can be run by any Retrieve user with Read access to the protocol referenced by the query. Public queries can be created only by users with Publish access to the protocol. *See also* private query, query, query library.

PXFR_RECV

A system account that contains all the protocol metadata when the protocol is imported.

PXFR_SEND

A system account that contains all the protocol metadata when the protocol is exported.

Q

query

A statement composed of conditions that identifies data that you want to retrieve from database tables.

Query By Form (QBF)

A query method in Retrieve that allows you to create a query using the online study pages you use for interactive data entry. *See also* query.

Query By Panel (QBP)

A query method in Retrieve that allows you to create a query using a graphical image of installed panels from which you can build a SQL statement that includes simple joins, outer joins, WHERE clauses, operators, and functions. *See also* query.

Query By SQL

A query method in Retrieve that allows you to create a query using Structured Query Language (SQL). You can use Query By SQL to create SELECT and DESCRIBE statements. *See also* query.

query library

A collection of queries created in Retrieve and saved to the database for future use. The queries in the query library are stored in an Oracle database table in the Clintrial software database. *See also* private query, public query, query.

query results

Records from database tables that meet the conditions specified by a query. *See also* query, record.

query results destination

The location in which records retrieved by a query are stored. The results destination can be a window on the computer monitor, an Oracle database table, or a SAS Data file. *See also* query, query results.

R

range checking

The process of applying tests to items to determine if data values fall within a defined range of values having an upper and a lower bound. In data entry, range checking occurs at data entry. In batch loading, range checking occurs during screening. *See also* batch loading, screening.

recoding

For an item that has been successfully coded, the process of replacing the current code with a different code.

reconciliation

The process of associating imported codelists, items, or flags and notes with the appropriate corresponding objects in the database instance where you are importing. *See also* import.

record

The data stored in one row in a database table. Depending on the panel type, the data collected in a study page is stored in one or more records in the database. A database table can contain one record per subject, multiple records per subject, one record per subject visit, or multiple records per subject visit. For example, the data in a study page that contains repeating items is stored in multiple records in the database.

record status (status)

A status that the Clintrial software assigns to records to track records internally as they go through various stages of data management. For example, when an update table record is first entered, its status is 2 (Unverified). When it is verified, its status changes automatically to 1 (Verified). Record statuses in the audit table reflect the location of the record when it was modified or deleted, and the type of modification it underwent. For example, if you delete a record in the update table with a status of 1 (Verified), the record's status in the audit table is -41 (Verified and deleted from update table).

regrouping records

In batch loading, the process of grouping batch-loaded records into observations.

request

A method of seeking more information about verbatim text that cannot be coded automatically, and that results in an omission record.

repeating block

A single block that data-entry operators can use multiple times. The designer determines how many times a block can repeat in a study book.

For example, a study book may contain a block called EXAM that contains study pages to record data on medications, vital signs, and lab results. Anticipating that different subjects will need different numbers of exams, the designer could make EXAM a repeating block. The data-entry operator can then record results of subsequent examinations as needed in repeats of the EXAM block.

Repeats of a block all use the same block key value, and are distinguished by block repeat key values. The designer can either assign block repeat key values for repeats of the block, or allow these values to be entered during data entry. *See also* block, block repeat key value.

repeating item

An item for which multiple values can be entered within a page section. If one item in a page section is repeating, other items within the page section are also repeating.

repeating study page

A single study page that data-entry operators can use multiple times. The designer determines how many times a study page can repeat.

For example, a block may contain a study page called VITALS that records data on a subject's vital signs. Anticipating that vital signs will need to be recorded several times during a visit for some subjects, the designer can make VITALS a repeating study page. The data-entry operator can then record results of subsequent readings as needed in repeats of the VITALS study page.

Repeats of a study page all use the same page key value, and are distinguished by page repeat key values. The designer can either assign page repeat key values for repeats of the study page, or allow these values to be entered during data entry. *See also* case report form (CRF), study page, page repeat key value.

replication

In Multisite, the movement of data in a protocol or account among multiple sites.

required item

See mandatory item.

revision control

A set of features in Design that help you to enforce metadata consistency throughout the protocol hierarchies that you create.

rule

Part of a PL/SQL statement that is attached to a panel, and used to confirm that clinical data meets the requirements of the clinical protocol. For example, a rule can confirm that all subjects meet a minimum age requirement of 18. Rules become part of the validation procedure that validates clinical data before it is moved to the data table. During validation, rules evaluate to TRUE or FALSE. *See also* rule action, validation procedure.

rule action

A part of a rule definition that determines the status of a record when a rule applied to that record evaluates to FALSE. A rule action can be either REPORT or REJECT. If the rule action is REPORT, the rule evaluated to FALSE, but the record will pass validation. If the rule action is REJECT, the rule evaluated to FALSE, and the record will fail validation. *See also* rule, validation procedure.

S

SAS Data file

A collection of files that contain the records retrieved by a query in a format that can be imported into SAS for analysis. The SAS Data file consists of an ASCII text file (*.dat) and a SAS command file (*.sas). Optionally, the SAS Data file can also contain a SAS proc format file (*.fmt). *See also* SAS proc format.

SAS format library

A file that contains SAS proc format statements for all Clintrial software codelists in a protocol. Proc format statements are used in SAS to decode Clintrial software data. *See also* SAS proc format.

SAS proc format

A file that contains proc format statements, which are used in SAS to decode Clintrial software data. *See also* SAS Data file, SAS format library.

screening

The process of updating system items, grouping the records into observations (if grouping records are specified in Design), and applying data checks to data that has been batch loaded into the update table. The data checks include confirming that values are provided for mandatory items, and applying the range checks and checklist tests that are defined for those items. Records that have been batch loaded and screened appear to be identical to those records entered interactively. *See also* batch loading, range checking, mandatory item.

searchlist

A list of protocols from which you can copy Clintrial software object definitions into the current protocol. *See also* parent protocol, protocol hierarchy.

sequence

A set of predefined default values for a repeating item. Sequences are associated with items through the page section layout Design Sequence option. *See also* repeating item.

server restriction

A SQL statement used to limit the number of omission records displayed in Classify's Omission Browser by restricting the protocols, thesaurus protocols, panels, or other general criteria from which the omission records are derived.

SI unit

A standardized unit of measure from the International System of Units. *See also* International System of Units.

sorting item

An item used to sort records in a page section based on a Type 0, Type 2, or Type 4 panel. A sorting item can be used to sort records in ascending or descending order. *See also* grouping item.

source object

In revision control, an object from which one or more other objects have been copied with connections. *See also* revision control, destination object.

source protocol

A Lab Loader protocol into which you batch load lab data. You work with the lab data in this protocol to prepare it for transfer to the destination protocol. *See also* destination protocol.

special context item

A context item used as a key, alone or with another special context item, to uniquely identify clinical data for a particular subject, block, or study page. The special context items are subject item, block key item, block repeat key item, page key item, and page repeat key item. *See also* context item.

sponsor

The manufacturer or developer of the drug or biomedical device for which clinical data is collected; that is, your company.

sponsor note

An annotation that the sponsor attaches to clinical data, consisting of a predefined note category, note name, and note text. For example, a sponsor note might be "MISSING/Investigator confirmed measurement not taken." *See also* sponsor, investigator note.

SQL restriction

A SQL WHERE clause that is used to restrict the records you are working with to those that meet specified conditions. The SQL WHERE clause can consist of one or more valid SQL conditions connected by logical operators. For example, you can use a SQL restriction to select all records where AGE is greater than 30. *See also* view protocol, view restriction.

standardization

A method for ensuring consistency of data definitions and clinical data. For example, to standardize data definitions, you can define database objects that are common to all clinical trials sponsored by your company.

step types

The transformations and lookup steps that compose a thesaurus algorithm.

study book

In the Clintrial software, clinical data is recorded in study books – online representations of case report forms (CRFs). Each study book contains an ordered list of study pages, corresponding to the pages in a paper CRF. When you open a particular study book, you gain access to its study pages, which you can then work on and navigate as a set.

In addition to clinical data study books, the Clintrial software also includes enrollment study books, used to enroll study subjects, and non-subject study books, used to enter nonclinical data such as standard coding thesauruses or laboratory normal ranges. *See also* block, case report form, study page.

study page

A data-entry window within a study book. In clinical data study books, each study page corresponds to a page in a case report form (CRF). *See also* case report form (CRF), context page section, page section, study book.

subject

A human subject, often a patient, in a clinical trial for whom clinical data is entered in the Clintrial software.

subject item

A subject-related context item used in enrollment that uniquely identifies the subject in the clinical data tables in a study. Typically, this is the subject identifier, but it can be any subject-related context item. The designer must specify the subject item as a protocol attribute for a clinical data protocol. *See also* context item, enroll, special context item.

subject list

A list of subjects that you can create to navigate a study book. For example, suppose that you usually enter data for subjects 100, 78, and 149, in that order. You can define a subject list that will display study pages for those subjects in that order. *See also* study book, study page, subject.

subset codelist

An Oracle view onto a base codelist. By using an optional subset restriction clause, a subset codelist can make available only certain codes from the base codelist. *See also* codelist, subset value.

subset key item

An item that is the key for uniquely identifying the records in a subset page section. The value of the subset key item is determined when the subset page section is placed on a page template; this value cannot be modified during data entry or editing. *See also* subset page section, subset key value.

subset key value

The value assigned to the subset key item when the subset page section is placed on a page template; this value cannot be modified during data entry or editing. The subset key value must be unique and must consist of a maximum of two characters. Subset key values can also be used to create within-panel master-detail relationships among page sections on a page template. *See also* subset key item, subset page section.

subset page section

One of a set of a page sections based on a Type 0, Type 2 or Type 4 panel that can occur multiple times on a study page, with each different value of the subset key item representing distinct rows (subsets) of data. Each occurrence of a subset page section constitutes a different observation. The subset page sec-

tions do not need to be the same. For example, they can include different items from the panel. *See also* subset key item, subset key value.

subset value

A numeric value associated with a subset of code values in a codelist that serves as the base codelist for subset codelists. A subset value can be used in a SQL restriction clause to create a subset codelist. *See also* subset codelist.

support elements

The characters, words, and phrases that certain transformations require. For example, if the purpose of a transformation is to remove words, you must define support elements that indicate which words to remove.

synonyms view

The thesaurus view that stores standard codes for clinical events, and descriptions that you consider synonymous with the standard descriptions in the terms view. There may be multiple synonyms views (such as company-approved, project-specific, and interim) for each coding thesaurus protocol.

system account

An Oracle account for internal Clintrial software use. For example, the CTS account stores information about Clintrial software protocols.

system item

Items internal to the Clintrial software that are attached automatically to the start of every record in the Clintrial software. For example, some system items are the subject ID (SUBJECT_ID), record status (STATUS), and the entry ID of the user who entered or last edited the current record in the database (ENTRY_ID). The meanings of system items may depend on the type of database table (update, data, or audit).

System Organ Class (SOC)

The highest level of the MedDRA hierarchy that provides the broadest concept for data retrieval.

system parameter

A parameter that defines the characteristics of the working environment for all users of an Oracle database instance to which the users connect through the Clintrial software. For example, the PASSWORD_MINIMUM system parameter sets the minimum password length for all Clintrial software users. *See also* protocol parameter, user preference.

T

tab order

The order in which you move through the enterable fields in a study page, set by the designer.

tags audit table

An Oracle table that stores copies of flags and notes that have been modified or deleted while attached to clinical data. There is one tags audit table for each protocol. *See also* auditing.

tags table

An Oracle database table that stores the flag and note data associated with clinical data. There is one tags table for each protocol. *See also* flag, note.

term

The standard, accepted words and phrases that describe a clinical event such as a disease or drug. In a coding thesaurus, a unique code is assigned to each term. A term can have one or more synonyms.

terms view

The thesaurus view that stores standard codes and their exact definitions. Every coding thesaurus protocol contains one and only one terms view.

text field

In data entry, a field in which you can enter any combination of letters, numbers, punctuation marks, spaces, or special characters (with the data type TEXT).

thesaurus account

See coding thesaurus protocol.

thesaurus algorithm

A sequence of steps that determines the most appropriate code match for the verbatim text (that is, text entered by the user), such as a disease or a drug name. The Clintrial software supplies a default thesaurus algorithm. You can create a customized thesaurus algorithm. *See also* coding, coding thesaurus.

thesaurus language

A language name that determines which language-specific text items in the terms, synonyms, and stopwords panels are used for coding. The Clintrial software uses the language name to select a corresponding language-specific text item that contains the language name. *See also* coding.

thesaurus protocol

See coding thesaurus protocol.

thesaurus view

The Oracle view onto a panel in a coding thesaurus protocol. A thesaurus view is created automatically for each panel added to a coding thesaurus protocol.

transfer map

Indicates the data to be transferred from a Lab Loader source protocol to a clinical data destination protocol by establishing a direct connection from the source items to the destination items.

transformation

A specific type of change made to verbatim text by a step in a thesaurus algorithm. Transformations standardize the format of text and remove or replace extraneous words and characters; they do not change the verbatim text's meaning.

U

unaggregated codelist

A codelist that is stored in its own Oracle database table. *See also* aggregated codelist, codelist.

unimplemented derivations

Derivations that have been modified within a panel that is marked for revision. The derivations are unimplemented if the revisions to the derivations have not yet been implemented.

unimplemented rules

Rules that have been modified within a panel that is marked for revision. The rules are unimplemented if the revisions to the rules have not yet been implemented.

update table

An Oracle database table that stores clinical data when it is first entered in the Clintrial software. The update table is a storage area for clinical data while it is being cleaned. You can edit, verify, and validate clinical data in the update table. After you have validated clinical data successfully, you can move the cleaned data from the update table to the data table. There is one update table for each installed panel. The name of the data table is *panel-name*_UPDATE; for example, LAB_UPDATE. *See also* data table, validation.

user account

An Oracle account to which a Clintrial software user connects to begin a work session. A user account must be granted access rights to the protocol account before the user can add, modify, display, or delete the data associated with that protocol.

usergroup

A list of Clintrial software users (that is, user accounts) to which you grant and revoke access rights as a group. *See also* access right.

user preference

Parameters that tailor the working environment for individual users. User preferences take their default values from corresponding system parameters. Each user account can change the default values for that account by changing the user preferences. Not all system parameters have corresponding user preferences.

For example, data-entry users can set a preference to automatically save data to the database when an open study page is closed. *See also* protocol parameter, system parameter, user account, usergroup.

user procedure

A PL/SQL procedure or package created by the designer or programmer for use by rules, derivations, and data-entry processing procedures. User procedures are also called customized functions or site-specific and protocol-specific functions.

V

validation

The process of running a validation procedure on clinical data to ensure that the data meets the requirements of the clinical trial for logical and consistent data. *See also* derivation, rule, validation procedure.

validation procedure

A PL/SQL procedure that is built automatically from derivations and rules associated with a panel. There is one validation procedure for each panel.

value changed procedure

A type of data-entry processing procedure that runs automatically when the Enter user changes the value of a field and then leaves the field. The designer attaches a value changed procedure to a field in a page section.

value field type

A field type that defines whether the value from a codelist is stored as text or as a number in the checklist item in a panel. The value field type must be of the same data type as the checklist item in the panel. *See also* checklist, codelist.

verbatim text

In coding, the entered text for which you want to assign a standard code from a coding thesaurus. *See also* coding, coding thesaurus.

verbatim text item

The item in a clinical data panel that stores verbatim text for a coding target. The Verbatim Text Item field is its corresponding study book field, accessed through Enter.

verification

The reentry of data to ensure that the data has been transcribed correctly. Some panels require verification before records can be validated and merged. There are two types of verification: interactive and blind verification. *See also* interactive verification, blind verification.

view codelist

A codelist that does not contain actual data but provides a view onto a Type 0 panel or a non-Clintrial Oracle database table. *See also* codelist, view protocol.

view panel

A view of the data in a panel that is part of a view protocol. *See also* checkpoint date, SQL restriction, view protocol.

view protocol

A protocol that does not contain actual data, but provides a view onto another protocol (that is, a base protocol). A view protocol displays records based on a specified date or a condition. *See also* base protocol, checkpoint date, view panel, view restriction.

view restriction

A SQL restriction clause on a view protocol that presents a select portion of clinical data that is in the base protocol. A view protocol with a view restriction might present only clinical data for subjects 100 through 500. *See also* Clintrial software protocol, view protocol.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

Clintrial 4.7.1
Reference Guide
Part Number: E27476