Oracle® Clinical

Application Programming Interface Guide

Release 5.1

E53565-02

March 2015

ORACLE®

Oracle Clinical Application Programming Interface Guide, Release 5.1

E53565-02

Contents

Preface

This manual describes the Oracle Clinical Application Programming Interface (API).

This preface contains the following topics:

Audience

This document is intended for skilled C and PL/SQL programmers.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Finding Information and Patches on My Oracle Support

Your source for the latest information about Oracle Clinical is Oracle Support's self-service Web site My Oracle Support (formerly MetaLink).

Before you install and use Oracle Clinical, always visit the My Oracle Support Web site for the latest information, including alerts, White Papers, installation verification (smoke) tests, bulletins, and patches.

Creating a My Oracle Support Account

You must register at My Oracle Support to obtain a user name and password account before you can enter the Web site.

To register for My Oracle Support:

1. Open a Web browser to https://support.oracle.com.
2. Click the **Register** link to create a My Oracle Support account. The registration page opens.
3. Follow the instructions on the registration page.

Signing In to My Oracle Support

To sign in to My Oracle Support:

1. Open a Web browser to https://support.oracle.com.
2. Click **Sign In.**
3. Enter your user name and password.
4. Click **Go** to open the My Oracle Support home page.

Finding Information on My Oracle Support

There are many ways to find information on My Oracle Support.

Searching by Article ID

The fastest way to search for information, including alerts, White Papers, installation verification (smoke) tests, and bulletins is by the article ID number, if you know it.

To search by article ID:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Locate the Search box in the upper right corner of the My Oracle Support page.
3. Click the sources icon to the left of the search box, and then select **Article ID** from the list.
4. Enter the article ID number in the text box.
5. Click the magnifying glass icon to the right of the search box (or press the Enter key) to execute your search.

   The Knowledge page displays the results of your search. If the article is found, click the link to view the abstract, text, attachments, and related products.

Searching by Product and Topic

You can use the following My Oracle Support tools to browse and search the knowledge base:

- Product Focus — On the Knowledge page under Select Product, type part of the product name and the system immediately filters the product list by the letters you have typed. (You do not need to type "Oracle.") Select the product you want from the filtered list and then use other search or browse tools to find the information you need.
- Advanced Search — You can specify one or more search criteria, such as source, exact phrase, and related product, to find information. This option is available from the **Advanced** link on almost all pages.

Finding Patches on My Oracle Support

Be sure to check My Oracle Support for the latest patches, if any, for your product. You can search for patches by patch ID or number, or by product or family.

To locate and download a patch:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Click the **Patches & Updates** tab. The Patches & Updates page opens and displays the Patch Search region. You have the following options:
   - In the **Patch ID or Number** field, enter the number of the patch you want. (This number is the same as the primary bug number fixed by the patch.) This option is useful if you already know the patch number.
   - To find a patch by product name, release, and platform, click the **Product or Family** link to enter one or more search criteria.
3. Click **Search** to execute your query. The Patch Search Results page opens.
4. Click the patch ID number. The system displays details about the patch. In addition, you can view the Read Me file before downloading the patch.
5. Click **Download.** Follow the instructions on the screen to download, save, and install the patch files.

Finding Oracle Documentation

The Oracle Web site contains links to all Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences Documentation

To get user documentation for Oracle Health Sciences applications, including an HTML version of this guide, go to the Oracle Health Sciences documentation page at:

http://www.oracle.com/technetwork/documentation/hsgbu-154445.html

---

Note: Always check the Oracle Health Sciences Documentation page to ensure you have the latest updates to the documentation.

Finding Other Oracle Documentation

To get user documentation for other Oracle products:

1. Go to the following Web page:

   http://www.oracle.com/technology/documentation/index.html

   Alternatively, you can go to http://www.oracle.com, point to the Support tab, and then click **Documentation.**
2. Scroll to the product you need and click the link.
3. Click the link for the documentation you need.

Related Documents

This section lists the documents in the Oracle Clinical documentation set, followed by their part number. The most recent version of each guide is posted on the Oracle Web site; see "Finding Oracle Documentation" on page ix.

- *Oracle Clinical Installation Guide*
- *Oracle Clinical Administrator's Guide*
- *Oracle Clinical Getting Started*
- *Oracle Clinical Creating a Study*
- *Oracle Clinical Conducting a Study*
- *Oracle Clinical Application Programming Interface Guide*
- *Oracle Clinical Remote Data Capture Onsite Administrator's Guide*
- *Oracle Clinical Remote Data Capture Onsite User's Guide*
- *Oracle Clinical Remote Data Capture Classic Data Entry User's Guide*
- *Oracle Clinical, Oracle Clinical Remote Data Capture, and Oracle Thesaurus Management System Security Configuration Guide*

The release notes (Article ID 1931214.1) and the release content document are available on My Oracle Support. See Finding Information and Patches on My Oracle Support for more information.

In addition, Oracle Clinical customers can request a copy of the *Oracle Clinical Stable Interface Technical Reference Manual* from Oracle Support.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Customizing Oracle Clinical

This section includes:

1

Custom Workflows, Menus, and Image Handling

The Oracle Clinical data entry process can be integrated with external workflow and imaging systems. Oracle Clinical communicates with the external system through a library of functions (RXCLBCLI.PLL) shipped with the product (in the admin directory on the CD-ROM) that you can customize. Also shipped with the functions is a set of menu options that you can enhance or supersede to modify the standard Oracle Clinical menus.

The Oracle Clinical functions that support integration with an external system—demonstrated here by an imaging workflow system—are on the product CD-ROM in the admin directory as a library. RXCLBCLI.PLL is the basis of the API. These integrating functions can be programmed in the library, using OLE, C, PL/SQL, or DDE. The PLL shipped with Oracle Clinical contains code that can be activated to see the simple workflow system used to test the functionality.

Also provided are five menu stubs—Custom Program Units— that can be customized for use with the external system. This is done by naming and displaying the menu items, and to writing the code to fully implement them.

Note: The existing Oracle Clinical function Get Image, which can be invoked from Data Entry, Discrepancy Database, and from the DCF, has been moved to the PL/SQL library. This enables the integration of the external flow with Oracle Clinical's Data Entry System. This external flow need not necessarily be an imaging system.

Usage Overview

Specific functionality is determined by the customizations that are made, so the following description provides a generic view of how Oracle Clinical and the external system might interact.

Entering Data for the Workflow Option

When the workflow option is enabled (see "Enabling External Systems" on page 1-4), the Data Entry menu shows the workflow task; otherwise, this task is hidden. All top menu items in the Data Entry tree are considered tasks. Before the task sequence is described, two basic terms, workitem and document, need to be defined.

A **workitem** is a collection of DCIs, also referred to as documents. All DCIs in the workitem are for the same task and the same study, but they may be for different patients. Some examples of tasks are Initial Log-In, and Second Pass Entry.

A **document** is a single DCI within a workitem, and can be composed of one or more physical Case Report Forms (CRF) pages.

To begin data entry, select Get Next Workitem from the Workflow menu; queries are not permitted. Oracle Clinical presents the DCIs according to the workitem sequence definition. When you commit a document, the next document in that workitem is immediately presented. If a problem prevents you from committing that document, you must select Get Next Document from the Workflow menu. When all documents in a workitem are completed, you can start the next workitem or close the workflow.

When you are performing data entry through the workflow system the following standard Oracle Clinical data entry options are unavailable from a data entry form:

- changing studies
- changing tasks
- turning DCI book on
- entering or executing a query

The key change task is also not available for use in a workflow system.

Working from the Discrepancy Management System

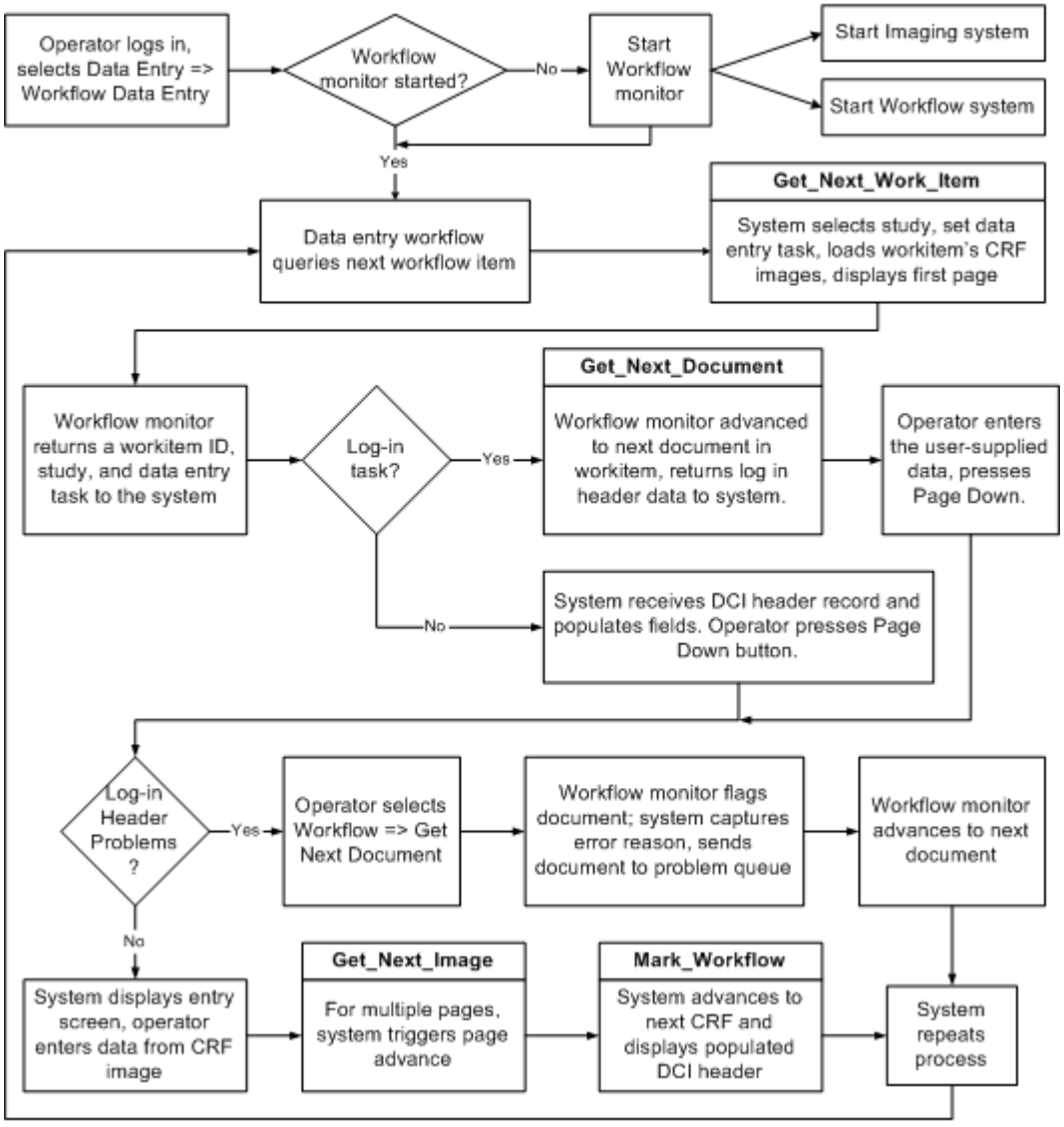
Once you enter the discrepancy database from the Conduct menu options and query discrepancies, you can use the workflow monitor system to invoke the image of the page where the discrepant question appears. You can then use standard Oracle Clinical functionality to handle discrepancies. For example, you can move to data entry to correct data or change review status before creating a Data Clarification Form (DCF).

Working with DCFs

You can use the external systems in managing and resolving discrepancies returned from investigators through the standard DCF. Assuming that the DCF has been scanned and indexes for the document based on DCF-ID, patient, site, investigator, you can use the imaging workflow option for performing the discrepancy resolution. One scenario to do this is:

1. Log in to Oracle Clinical.
2. Select the DCF screen.
3. Query for a DCF with the RECEIVED status.
4. Invoke the image corresponding to the current DCF.
5. Use standard Oracle Clinical functionality to move to the Discrepancy Database for the discrepancy entries on the DCF form.
6. Choose to display a Data Entry form via the Special in-form menu.
7. Invoke the corresponding CRF's image in a separate window.

Figure 1–1 CRF Entry Workflow diagram



This diagram graphically shows the workflow for entry of CRFs. The image crf_entry_workflow_diag.gif shows shows the workflow for entry of CRFs. This description breaks this process down into the following general steps:

1. The operator logs in and selects Data Entry, then Workflow Data Entry.

2. If the Workflow monitor is not started, the operator must start it, then start the Imaging system and Workflow system before proceeding. If the Workflow monitor is started, the data entry workflow queries the next workflow item by calling the procedure Get_Next_Work_Item.

3. Get_Next_Work_Item prompts the system to select the study, set the data entry task, load the workitem's CRF images, and displays the first page. The workflow monitor then returns a workitem ID, study, and data entry task to the system.

4. If this is a Log-in task the system calls the Get_Next_Document procedure, which prompts the system to advance the workflow monitor to the enxt document in the workitem and returns the log in the header data to the system. After this, the operator,

enter the user-supplied data. If this is not a Log-in task, the system receives the DCI header record and populates fields. In either case, the operator then proceeds by pressing Page Down.

5. If problems exist in the Log-in Header, the operator selects Workflow, then Get Next Document. The workflow monitor then flags the document, and the system captures the error reason and sends the document to the problem queue. Proceed to step 7.

6. If no problems exist in the Log-in header, the system displays the entry screen and the operator enters data from CRF image. The system then calls the Get_Next_Image procedure, which triggers page advance for multiple pages. The system then calls the Mark_Workflow procedure, which prompts the system to advance to the next CRF and displays populated DCI header.

7. The system repeats the process by returning to step 2, as the data entry workflow queries the next workflow item.

End of description.

****

Enabling External Systems

To enable an external workflow system, set the INVOKE_WORKFLOW option to Y in the OCL_STATE reference codelist. To enable an imaging system, you set the INVOKE_IMAGE option to Y in the OCL_STATE reference codelist. To enable these options:

1. Select **Admin**, then **Reference Codelists**, and **Local Codelists**.
2. Type OCL_STATE in the **Name** field, and execute a query.
3. To enable an external workflow system, scroll through the values to INVOKE_WORKFLOW.

   To use an imaging system, scroll through the values to INVOKE_IMAGE.
4. Set the value to Y.
5. Save the data.

Menu Customization

Oracle Clinical provides functions that allow for two levels of customization. These are modifying the menus and implementing API functions.

When Oracle Clinical opens a form, it calls the wnfi procedure from the PL/SQL library. This enables you to set properties for the menu you are customizing. It can be used to update labels for the customizable options, or even to disable them. The procedure itself contains samples.

Other functions customize the Get Next and Get Previous Image options, which are workflow tasks. They allow you to proceed to the next or previous page of a multi-page document from within Oracle Clinical, when you are viewing an image of the document in conjunction with using data entry.

In rxclbcli.pll, there are five functions that are defined to allow you to add your own menu options and functionality to the Data Entry menu. The functions, are named CustomPgmUnit1 through CustomPgmUnit5.

The CustomPgmUnit# functions are provided for each custom menu option as a shell. You can add functionality into the data entry system. The return value from these

functions is either a null or an error message. If the function returns an error message the value of the error_type is checked, a value of 1 causes a warning message to be displayed. If the error_type's value is 2, an error message is displayed and an error is raised from the calling form.

These functions have arguments that are used to pass data into the function or return data from the function. In the tables describing the arguments the In or Out column has an I for input argument and an O for an output argument.

In the tables the Data types column shows the defined type of data for the variable. Sometimes the data type is defined to be the same as a particular column in a database table. The format that shows this is the database table name, period, column name, followed by % type. For example **received_dcis.document_number%type** means the data type is defined as the same type used to define the column document_number in the received_dcis table.

CustomPgmUnit1

CustomPgmUnit1(*document_number*, *nreceived_dci_id*, *nError_type*)

Argument	In or Out	Data type
document_number	I	received_dcis.document_number%type
received_dci_id	I	received_dcis.received_dci_id%type
nError_type	O	number

CustomPgmUnit2

CustomPgmUnit2(*ndcf_id nError_type*)

Argument	In or Out	Data type
ndcf_id	I	data_clarification_forms.dcf_id%type
nError_type	O	number

CustomPgmUnit3

CustomPgmUnit3(*nError_type*)
nError_type out number

Argument	In or Out	Data type
nError_type	O	number

CustomPgmUnit4

CustomPgmUnit4(*nError_type*)

Argument	In or Out	Data type
nError_type	O	number

CustomPgmUnit5

CustomPgmUnit5(*nError_type*)

Argument	In or Out	Data type
nError_type	O	number

Some Possible Ways to Work with the API

The target file is opened depending on the application associated with it. The system prompts for an application name if one is not given.

- Create a text file with the TEXT_IO command on a file server accessible to the client, and then use web.show_document ('file://...') to access the file. The file is opened with the application associated with the extension of the file on the client PC. When no associated application is named, the system prompts for one. An example of this file protocol method is in the Get Image procedure.
- Create a text file on a Web server with the TEXT_IO command and access the file with web.show_document ('http://...'). The file is opened with the application associated with the extension of the MIME type that is associated with the file on the Web server. When no associated application is named, the system prompts you. The Get Image procedure has an example of this method.

  For Oracle WebDB listener, you can view, and add, MIME type extensions through the MIME TYPE option under the Listener mode.
- Follow a similar method to the preceding one, but create the text file in the same directory as the file `opabasejini.htm`, specifying the name of the Web server. The file `opabasejini.htm` is stored in *oracle_home*/html. The value of *oracle_home* can be derived by using tool_env.getvar('oracle_home',...). Then, use web.show_document ('http://...'). The advantage of this method is that you do not have to create another virtual directory, and you do not have to specify the virtual directory in web.show_document.
- Call a form that starts up a Java bean.

Workflow API Functions

Plan the functions you need from the following sections. Oracle Clinical calls the functions at specific execution points. Functions created by customers will be included in the forms library dedicated to user-defined program units (PL/SQL functions or procedures).

These functions execute at the client level so that communication mechanisms between Oracle Clinical and the workflow monitor are customizable. When the workflow system is invoked, on return, there is a workitem opened for the specified userid.

Arguments are used to pass data into the functions and/or return data from the functions. In the tables describing the arguments the In or Out column has an I for input argument, O for an output argument, and I/O for both an input and output argument.

In the tables the Data types column shows the defined type of data for the variable. Sometimes the data type is defined to be the same as a particular column in a database table. The format that shows this is the database table name, period, column name, followed by % type. For example **oracle_accounts.oracle_account_name%type** means the data type is defined as the same type used to define the column oracle_account_name in the oracle_accounts table.

The API function that you write generally return a string. A NULL string value indicates successful execution. If your function returns a string that is not NULL the

Oracle code treats this as an error; the workflow stops processing, the string value is an error message and is displayed.

get_next_CRF_work_item

`get_next_CRF_work_item (userid, clinical_study_id, task, work_item_id, status)`

Argument	In or Out	Data type
userid	I	oracle_accounts.oracle_account_name%type
clinical_study_id	I/O	clinical_studies.clinical_study_id%type
task	I/O	varchar2
work_item_id	O	number
status	O	varchar2

This routine is called from the Special menu item of the Log-In form. The function returns a null value to indicate successful completion. Otherwise, it returns an error message.

The task returned by the function should be one of the following values:

INITIAL LOG-IN
INITIAL LOG-IN AND FIRST-PASS ENTRY
FIRST-PASS ENTRY
SECOND-PASS ENTRY
RECONCILIATION
UPDATE
BROWSE

The status code returned by the function should be SUCCESS or EOF. When the status code is EOF it indicates that there are no work items to process. A message indicating this is displayed.

Execution of this routine enables the following:

- The workflow monitor starts if this module is not yet started. (The workflow monitor starts the image viewer, if necessary.)
- The workflow monitor retrieves the next workitem from the workflow system for the specified user.
- The Data Entry task (for example, PASS 1 ENTRY), the clinical_study_id, and the work_item_id are returned to Oracle Clinical. The work_item_id is stored for use by other client API routines.
- Oracle Clinical initializes the task and sets study context, then calls the `get_next_document` function.

get_next_document

`get_next_document (source, task, work_item_id, CRFpage, document_number, patient_number, clinical_planned_event, DCI_short_name, RDCI_date, RDCI_time, subevent_number, status)`

Argument	In or Out	Data type
source	I	varchar2

Argument	In or Out	Data type
task	I	varchar2
work_item_id	I	number
CRFpage	O	dci_book_pages.start_page_number%type
document_number	O	received_dcis.document_number%type
patient_number	O	patient_positions.patient%type
clinical_planned_event	O	received_dcis.clin_plan_eve_name%type
DCI_short_name	O	dcis.short_name
RDCI_date	O	received_dcis.dci_date%type
RDCI_time	O	received_dcis.dci_time%type
subevent_number	O	received_dcis.subevent_number%type
status	O	varchar2

The purpose of this function is to retrieve information about the next workflow work item. The function returns a null value if it succeeds and an error message if it fails.

This routine is called either automatically—after get_next_workitem or when it has completed and is going to retrieve the next document in the current workitem—or manually. It is called in the following circumstances:

- whenever INITIAL LOG-IN, INITIAL LOG-IN AND ENTRY, PASS 1 ENTRY, PASS 2 ENTRY, UPDATE, BROWSE or COMPARISON RECONCILIATION is first invoked and a new workitem is being initialized (SOURCE=AUTO)
- whenever the next document in the current workitem is required (SOURCE=AUTO)— that is, when a new received DCI is to be created or an existing received DCI is to be retrieved
- whenever the get_next_document function is called from the Workflow menu (SOURCE=MANUAL)

The arguments for this function are:

- *CRF_page_number*: Number of the first page in the document.
- *document_number*: The unique number assigned by the workflow system for the document. The Log-In tasks fills the Oracle Clinical Document Number field with the contents of this variable. (On data entry or Browse tasks, this field is used to set the Oracle Clinical document field to retrieve the received DCI record for entry.)
- *patient_number*: The Oracle Clinical patient number, entered into the workflow system as part of the indexing operation. The Log-In tasks fills the Patient field in the Oracle Clinical DCI header with the contents of this variable. It is ignored in data entry or Browse operations.
- *clinical_planned_event*: The Oracle Clinical visit name, entered into the imaging system as part of the indexing operation. The Log-In tasks fills the Clinical Planned Event field in the Oracle Clinical DCI header with the contents of this variable.
- *DCI_short_name*: The DCI Name. The Log-In tasks fills the DCI Name in the Oracle Clinical DCI header with the contents of this variable. It is ignored in data entry or Browse operations. If the DCI Book Name field is null on entry, the value

returned is taken from the imaging record. If DCI Book Name is not null, the value is retrieved from the DCI book.

- *status*: SUCCESS or EOF.
  - SUCCESS causes processing of the returned document.
  - EOF indicates there are no more documents in this workitem. Oracle Clinical function displays a message indicating that there are no more documents to process. The user then can select the menu item to get a new workitem.

When this function returns and the current task is Initial Log-in (or Initial Log-in and Entry) Oracle Clinical populates the DCI header with the returned data and waits for the operator to create the received DCI (and move to data entry if in Initial Log-In and Entry).

If this function is called from the First Pass Data Entry, Second Pass Data Entry, Update, Browse, or Comparison Reconciliation task, Oracle Clinical uses the returned document_number to query the received DCI record while waiting for the operator to move to data entry.

It is possible that the document has already been logged into Oracle Clinical in error, so that it is not possible to log in the duplicate RDCI. In such a case, the function returns an error. The operator is expected to correct the error for example, by creating a new subevent or by moving the document to a workflow error queue.

It is possible that an attempt is made to enter data (Pass 1 or Pass 2) into a document not logged into Oracle Clinical. While this should be an extremely rare occurrence, if it happens, the operator should move the document to a workflow error queue. This action is accomplished, as above, by selecting Get New Document from the Workflow menu.

get_next_image

```
get_next_image (document_number)
```

Argument	In or Out	Data type
document_number	I	received_dcis.document_number%type

This user-written routine calls the workflow monitor and displays the image of the next page of the document. It is called when the submenu item Get Next Image is selected from the Workflow menu. The function returns a null value if it succeeds and an error message if it fails.

get_previous_image

```
get_previous_image (document_number)
```

Argument	In or Out	Data type
document_number	I	received_dcis.document_number%type

This user-written routine is called when the submenu item Get Previous Image is selected from the Workflow menu. It displays the image of the previous page of the document. The function returns a null value if it succeeds and an error message if it fails.

get_next_dcf_image

```
get_next_dcf_image (dcf_id)
```

Argument	In or Out	Data type
dcf_id	I	data_clarification_forms.dcf_id%type

This routine calls the workflow monitor and displays the image of the next page of the DCF. It is called when the submenu item Get Next Image is selected from the menu in the DCF form. The function returns a null value if it succeeds and an error message if it fails.

get_previous_dcf_image

```
get_previous_dcf_image (dcf_id)
```

Argument	In or Out	Data type
dcf_id	I	data_clarification_forms.dcf_id%type

This routine is called when the submenu item Get Previous Image is selected from the menu in the DCF form. It displays the image of the previous page of the DCF. The function returns a null value if it succeeds and an error message if it fails.

mark_workflow

```
mark_workflow (document_number, received_dci_id, received_dci_status_code)
```

Argument	In or Out	Data type
document_number	I	received_dcis.document_number%type
received_dci_id	I	received_dcis.received_dci_id%type
received_dci_status_code	I	received_dcis.received_dci_status_code%type

This routine is called at the end of the commit process, which occurs whenever changes are committed to the database. The workflow monitor updates its tables with the information it receives and moves the document along in the workflow process. Information received includes visit date, status, and other information from the Oracle Clinical received DCI record.

The execution point in Oracle Clinical where this routine is called is whenever a new received DCI has been committed to the database (Initial Log-In) or when Pass 1 or Pass 2 data has actually been saved to the database after data entry.

The function returns a null value if it succeeds and an error message if it fails. In the latter case the returned error message is displayed by the calling form as a warning.

close_workflow

```
close_work_flow ()
```

This routine is called to shut down the workflow monitor and imaging systems. This function is called when Oracle Clinical is closed, or on command from the Special submenu menu item. The function returns a null value if it succeeds and an error message if it fails.

getimage

getimage (*Document_number*, *Study_id*, *Study*, *Received_dci_id*, *Received_dci_id_entry_ts*, *Dci_id*, *Dci_name*, *Dci_clin_plan_eve_id*, *Dci_visit_number*, *Dci_clin_plan_eve_name*, *Dci_subevent_number*, *Received_dcm_id*, *Received_dcm_id_entry_ts*, *Dcm_id*, *Dcm_name* , *Dcm_clin_plan_eve_id*, *Dcm_visit_number*, *Dcm_clin_plan_eve_name*, *Dcm_subevent_number*, *Patient_position_id*, *Patient*, *Site_id* , *Site*, *Investigator_id*, *Investigator*, *User_id*, *User_password* , *Owning_location*, *Current_location*, *Env_type*, *CrfPageNo*)

Argument	In or Out	Data type
Document_number	I	received_dcis.document_number%type
Study_id	I	received_dcis.clinical_study_id%type
Study	I	clinical_studies.study%type
Received_dci_id	I	received_dcis.received_dci_id%type
Received_dci_id_entry_ts	I	received_dcis.received_dci_entry_ts%type
Dci_id	I	dcis.dci_id%type
Dci_name	I	dcis.name%type
Dci_clin_plan_eve_id	I	received_dcis.clin_plan_eve_id%type
Dci_visit_number	I	received_dcis.visit_number%type
Dci_clin_plan_eve_name	I	received_dcis.clin_plan_eve_name%type
Dci_subevent_number	I	received_dcis.subevent_number%type
Received_dcm_id	I	received_dcms.received_dcm_id%type
Received_dcm_id_entry_ts	I	received_dcms.received_dcm_entry_ts%type
Dcm_id	I	received_dcms.dcm_id%type
Dcm_name	I	dcms.name%type
Dcm_clin_plan_eve_id	I	received_dcms.clin_plan_eve_id%type
Dcm_visit_number	I	received_dcms.visit_number%type
Dcm_clin_plan_eve_name	I	received_dcms.clin_plan_eve_name%type
Dcm_subevent_number	I	received_dcms.subevent_number%type
Patient_position_id	I	received_dcis.patient_position_id%type
Patient	I	received_dcis.patient%type
Site_id	I	received_dcis.site_id%type
Site	I	received_dcis.site%type
Investigator_id	I	received_dcis.investigator_id%type
Investigator	I	received_dcis.investigator%type
User_id	I	oracle_accounts.oracle_account_name%type
User_password	I	encrypted_passwords.password%type
Owning_location	I	clinical_studies.owning_location%type
Current_location	I	clinical_studies.owning_location%type
Env_type	I	varchar2

Argument	In or Out	Data type
CrfPageNo	I	dci_book_pages.start_page_number%type

GetImage invokes the external imaging system. Oracle Clinical passes the context information for the current RDCI. It is invoked by Special Menu Item, F3 key or the getimage icon on Log-In and Discrepancy forms.

getImageDCF

```
GetImageDCF(dcf_id, user_id, user_password)
```

Argument	In or Out	Data type
dcf_id	I	data_clarification_forms.dcf_id%type
user_id	I	oracle_accounts.oracle_account_name%type
user_password	I	encrypted_passwords.password%type

GetImageDCF invokes the external imaging system from the DCF screen. Oracle Clinical passes in dcf_id, user_id and password. It is invoked by Special Menu Item, F3 key or the get_image icon on Log-In and Discrepancy forms.

wnfi

```
wnfi (error_type)
```

Argument	In or Out	Data type
error_type	O	number

This routine is used to customize menu item labels. It also can be used to enable or disable the display of menu items. This routine is called when either the Log-In form (RXCDEMLI), the Data Entry form (RXCDECDE), the Discrepancy Management form (RXCDMMDD), or the DCF form is opened. The wnfi function returns a string.

The function returns a null value if it succeeds and an error message if it fails. In the latter case the error_type is checked. If the error_type is 1, a warning message, corresponding to the returned value is displayed. If the error_type is 2, the error message is displayed and an error would be raised from the calling form.

webpasskeys

```
webpasskeys(URL, Document_number, Study_id, Study, Received_dci_id, Received_dci_
id_entry_ts, Dci_id, Dci_name, Dci_clin_plan_eve_id, Dci_visit_number, Dci_clin_
plan_eve_name, Dci_subevent_number, Received_dcm_id, Received_dcm_id_entry_ts,
Dcm_id, Dcm_name ,  Dcm_clin_plan_eve_id, Dcm_visit_number, Dcm_clin_plan_eve_
name, Dcm_subevent_number, Patient_position_id, Patient, Site_id , Site,
Investigator_id, Investigator, User_id, User_password , Owning_location, Current_
location, Env_type, CrfPageNo)
```

Argument	In or Out	Data type
URL	I	varchar2
Document_number	I	received_dcis.document_number%type

Argument	In or Out	Data type
Study_id	I	received_dcis.clinical_study_id%type
Study	I	clinical_studies.study%type
Received_dci_id	I	received_dcis.received_dci_id%type
Received_dci_id_entry_ts	I	received_dcis.received_dci_entry_ts%type
Dci_id	I	dcis.dci_id%type
Dci_name	I	dcis.name%type
Dci_clin_plan_eve_id	I	received_dcis.clin_plan_eve_id%type
Dci_visit_number	I	received_dcis.visit_number%type
Dci_clin_plan_eve_name	I	received_dcis.clin_plan_eve_name%type
Dci_subevent_number	I	received_dcis.subevent_number%type
Received_dcm_id	I	received_dcms.received_dcm_id%type
Received_dcm_id_entry_ts	I	received_dcms.received_dcm_entry_ts%type
Dcm_id	I	received_dcms.dcm_id%type
Dcm_name	I	dcms.name%type
Dcm_clin_plan_eve_id	I	received_dcms.clin_plan_eve_id%type
Dcm_visit_number	I	received_dcms.visit_number%type
Dcm_clin_plan_eve_name	I	received_dcms.clin_plan_eve_name%type
Dcm_subevent_number	I	received_dcms.subevent_number%type
Patient_position_id	I	received_dcis.patient_position_id%type
Patient	I	received_dcis.patient%type
Site_id	I	received_dcis.site_id%type
Site	I	received_dcis.site%type
Investigator_id	I	received_dcis.investigator_id%type
Investigator	I	received_dcis.investigator%type
User_id	I	oracle_accounts.oracle_account_name%type
User_password	I	encrypted_passwords.password%type
Owning_location	I	clinical_studies.owning_location%type
Current_location	I	clinical_studies.owning_location%type
Env_type	I	varchar2
CrfPageNo	I	dci_book_pages.start_page_number%type

This routine is used to specify the location of the image to be displayed. Oracle Clinical passes the keys to allow the function to determine the appropriate image. The returned value is the specification string to access the image to be displayed.

An example of the use of this function is if getimage calls it to obtain the location of the image to be displayed and then displays the image.

is_web2

```
is_web2 ()
```

This function is no longer necessary. It was used to determine whether this was a Web environment or a client/server environment. It should always return TRUE.

2

Custom Identifier Handling

The OCL_CLIENT_PACK database package contains procedures that let you define object-naming validation for your Oracle Clinical installation. If you are the Oracle Clinical site administrator, you generally will write the function body to create site-specific validation and derivation code before you run the package. You can implement local naming conventions for:

- object names
- Case Report Form (CRF) document number validation
- document number derivation
- invoking a scanned image
- CRF page tracking

Overview

The OCL_CLIENT_PACK package consists of two files in the RXC_INSTALL directory, ocl_client_ps.sql and ocl_client_pb.sql. Refer to these files while reading this chapter. The files serve the following purposes:

- ocl_client_ps.sql
  - Package specification
  - Creating the public synonym
  - Granting execute privilege to both RXCLIN_MOD and RXCLIN_READ database roles
- ocl_client_pb.sql
  - Container for the function bodies for the OCL_CLIENT_PACK package

Use SQL*Plus from the RXC database account to execute `ocl_client_ps.sql`:

- when initially installing the Oracle Clinical application
- once after each time you modify the file

This file is a skeleton for creating custom validation and derivation because each function provides a stub; as delivered all validations succeeds and no derivation occurs. Each function includes detailed header comments that describe its purpose, parameters, and usage.

The functions in the OCL_CLIENT_PACK package are described in the following sections. The functions to customize for validation and derivation are ValidateName, ValidateDocument, and DeriveDocumentNumber. The functions to customize for page

tracking are DefaultBookPageNumber, TrimPageNumber, ChangePageStatus, and AddPhysPageNumber. In each case, the function duplicates a function that already exists in the Oracle Clinical code. If you write new code for the function, your code supersedes the existing application code; otherwise, it simply uses the default system code. The SQL files ocl_client_ps.sql and ocl_client_pb.sql in the RXC_INSTALL are well commented, use them for additional information and to implement your custom functions.

ValidateName

This function is called by the validation triggers on the naming of each of the Oracle Clinical objects. These objects include discrete value groups, questions, question groups, DCMs, DCIs, Procedures, Where clauses, and queries. As input, the function takes a code for the type of object and the name being validated. If the validation succeeds, the function returns NULL and the user can continue.

If the validation fails, the function returns a varchar2 character string, that contains an error message. The error message is displayed in the status line of the form. This function is called by the following forms that validate the name of the object being created. You can add your own validation to enforce your company naming standards as an example:

- Maintain DVG
- Maintain Questions
- Maintain Question Groups
- Maintain DCMs
- Maintain DCIs
- Maintain Procedures (both Derivation and Validation)
- Maintain Copy Groups
- Maintain Where Clauses
- Maintain Labs
- Mass Changes

ValidateDocument

This function is called by Validation Procedures on the document number in the CRF Log-In form and the Batch Data Load program. The input to the function is the document number being validated. If the validation succeeds, the function returns NULL; otherwise, the function returns the error message the user defined in the package.

DeriveDocumentNumber

With this function you can derive CRF document numbers from the key fields entered in the Log-In form and the Batch Data Load program. If you modify this function, you must review the argument list, reapply the modifications, then reinstall the package. Inputs are:

- **Key fields from the CRF and their ID numbers:** Study, Clinical_Study_ID, Patient, Patient_Position_ID, Investigator, Investigator_ID, Site, Site_ID, DCI, DCI_ID, Event, Clin_Plan_Eve_ID, and Subevent

- **Indicator of test or production mode date:** RXC_ENV_TYPE

The function returns NULL if the derivation is not possible; otherwise it returns the derived document number. Document numbers generated by this procedure are *not* validated against the ValidateDocument function; they are assumed to be valid.

DefaultBookPageNumber

This function assigns the correct page number to the first page of a DCI in the DCI book. The function returns zero if the operation is successful, or a non-zero error message number. The message number (message_topic_id) is used by the Maintain DCI Books to display the appropriate error message from the message_topics table. It uses the TrimPageNumber function and has these input parameters:

- first page number of the previous DCI in the DCI book
- numbering scheme of the previous DCI
- number of expected pages

The function's one output parameter corresponds to the default page number that represents the first page number of a DCI in the DCI book.

TrimPageNumber

This function trims the page number off its suffix according to the numbering scheme. It is called each time Oracle Clinical needs to add one unit to a page number and each time Oracle Clinical needs to assign a default page number.

There are two input parameters:

- page number to be trimmed
- numbering scheme according to which the page number needs to be trimmed

There are two output parameters:

- the result of the trimming operations
- Y when the other output parameter contains a numeric value; N otherwise

Examples of parameter values:

input 1	input2	output 1	output 2
2.1	NUMERIC	2	Y
10A	ALPHANUMERIC	10	Y
2.1	ALPHANUMERIC	2.1	N

ChangePageStatus

This function sets the client-specific status of a new page, depending on the current page status, the RDCI status, and the blank flag, according to user-defined rules. It returns NULL, if successful; otherwise, it returns an error message.

AddPhysPageNumber

This function adds a number to a page number, according to the numbering scheme—when DCIs are added to a DCI book, and also when pages assigned at data entry

correspond to one of the following unplanned events: the DCI is not in the DCI book; the DCI is present in the DCI book but not for the specific visit; or the subevent number is zero.

There are three input parameters:

- page number to which the number must be added
- number to be added to the page number
- numbering scheme

There is one output parameter: the page number after the operation.

input 1	input2	input 3	output
2.1	3	NUMERIC	2.4
10A	3	ALPHANUMERIC	10D

Part II

Interfacing with the Design Subsystem

This section includes:

3

Interface Tables and Views

This section contains the following topics:

Overview

Oracle Clinical interfaces with your company's applications in the areas of study planning, investigator management, and drug supply. The interface provides tables and modules that support basic requirements in these areas. You can use these Oracle Clinical tables and modules at your site, or you can replace them with tables and modules from your system. Oracle Clinical provides the means for validating responses using your company's own thesaurus system.

The Oracle Clinical tables and modules that are part of the Interface Configuration interface with external applications. The table names begin with OCL_. You can use these Oracle Clinical tables and modules at your own sites; alternatively, you can choose the following methods for your interface:

- Synonyms to customer tables
- Views on customer tables
- Batch/parallel maintenance of the OCL_ tables from customer applications

Mapping can be limited to the tables and columns required by the core Oracle Clinical functionality and does not have to reproduce exactly the Oracle Clinical version of these tables, described in "Table Details" on page 3-5.

> **Note:** If you customized the interface then do not maintain these tables with Oracle Clinical functions.

Pay particular attention to performance when you test your modifications. This observation has special importance when views have contributed to the mapping, because they can seriously affect performance. Also, the underlying structure and indexing of the customer application may not be tuned to meet Oracle Clinical access requirements.

The following sections describe the layout and content of each Oracle Clinical table and the required components that a customized interface supports.

Study Planning

Study planning is the process of controlling and managing all studies within a company and across all company locations. The standard procedure is to define studies only at the company location where they are to be conducted or analyzed. Study planning also contains information about studies that are to be run by outside organizations, such as CROs, as well as details of possible future studies.

Oracle Clinical requires that study planning provide:

- A two-tier hierarchy of studies to which users can be assigned as an alternative to maintaining security by individual user assignments to studies
- A list of studies that are approved for definition in Oracle Clinical

Table 3–1 Study Planning Tables

Table	Description
OCL_STUDIES	Master list of all studies
OCL_STUDY_REGIONS	List of regions where each study is to be conducted
OCL_ORGANIZATION_UNITS	List of company departments that can support studies
OCL_PROGRAMS	Upper tier where studies can be grouped and security managed
OCL_PROJECTS	Lower tier where studies can be grouped and security managed
OCL_PROGRAM_PRODUCT_MASTERS	List of the compounds for which each program is responsible

Drug Supply

A drug supply system interacts with Oracle Clinical in two ways:

- At the front end of the clinical process, the drug supply system provides a list of drugs formulated for a clinical study.
- At the back end of the process, the drug supply system takes the randomization generated in the design subsystem of Oracle Clinical and produces from it the packaged and labeled supplies for the study.

Table 3–2 Drug Supply Tables

Table	Description
OCL_DOSAGE_FORMS	Look-up list of dosage forms
OCL_PRODUCT_MASTERS	List of active substances (raw chemicals) and formulated products (medications) that a clinical study can use

Investigator Management

An investigator management system is responsible for maintaining a list of the investigators who can work on studies and the records of their assignments to studies.

Table 3–3 Investigator Management Tables

Table	Description
OCL_INVESTIGATORS	List of potential and actual investigators who can work on studies
OCL_SITES	List of locations where a clinical study can be conducted
OCL_STUDY_SITES	List of the assignments of sites to a particular clinical study
OCL_STUDY_SITE_ROLES	List of which investigators have been or are responsible for each study site

Thesaurus

The thesaurus, or dictionary, is used to validate responses to questions marked as requiring thesaurus validation. There are two types of dictionaries: Drug Names and Adverse Events.

Interface Procedures to Be Maintained by the Customer

The following procedures maintain referential integrity between Oracle Clinical and another customer application. You must maintain independence between Oracle-supplied code and customer applications, while maintaining referential integrity and complying with site support agreement rules. These requirements are provided for by a set of database procedures that can be maintained and used either by Oracle or by the customer.

The following sections list each test or update separately; in the actual implementation some are merged. Local procedures may differ, but the parameters and procedure names are fixed.

Prevent Study Deletion

Prevents a study from being deleted by checking in related tables in non-Oracle Clinical applications before a clinical study is deleted. Only a return value of OK allows the study to be deleted; anything else means failure.

```
LocalStudyDeletion.LocalDeletionOK
Param = nStudyId in number(10)
    vReturnMsg out Char(60)
```

Delete Interface Records

Deletes records from the OCL tables used by Oracle Clinical. A return value of DONE means that everything worked correctly; anything else means failure.

```
LocalStudyDeletion.DeleteOCLRecords
Param = nStudyId in number(10)
     vReturnMsg out Char(60)
```

Delete Local Records

Deletes clinical study-related records from non-Oracle Clinical tables when a clinical study is deleted. A return value of DONE means that everything worked correctly; anything else means failure.

```
LocalStudyDeletion.DeletelocalRecords
```

Param = nStudyId in number(10)
vReturnMsg out Char(60)

Prevent Region Deletion

Checks if a region can be deleted based on local usage. A return value of DONE means that everything worked correctly; anything else means failure.

LocalRegionUpdateDeletion.LocalRegDeletionOK
Param = nRegionId in number(10)
vRegionCode in char(7)

vReturnMsg out Char(60)

Update Region Code

Applies a cascade update when a region code is changed.

LocalRegionUpdateDeletion.LocalRegionCascadeUpdate
Param = vOldRegionCode in varchar2(7)
vNewRegionCode in varchar2(7)

vReturnMsg out varchar2(60)

Treatment Pattern Deletion Check

Checks if a treatment pattern can be deleted based on local usage. A return value of DONE means that everything worked correctly; anything else means failure.

LocalTreatPattDeletion.LocalTreatPattDeletion
Param = nTreatPattId in number(10)
vReturnMsg out char(60)

Treatment Assignment Deletion Check

Checks if a treatment assignment can be deleted based on local usage. A return value of DONE means everything worked correctly; anything else means failure.

LocalTreatAssnDeletion.LocalTreatAssnDeletionOK
Param = nTreatAssNum in number(10)
vStartCode in char(10)

vreturnMsg out char(60)

Get the Value of a Study Site Code

Supplies the value of a STUDY_SITE code.

LocalStudySite.GetStudySite
Param = vStudySite in varchar2 (10)
nStudyId in number(10)

vSiteCode in varchar2(10)

nSiteId in number(10)

vTestProd in varchar2(1)

vReturnStudySite out varchar2(10)

Validate a Study Site Code

Validates a STUDY_SITE code. A return value of OK means the value is acceptable.

LocalStudySite.ValidateStudySite
Param = vStudyCode in varchar2(10)
nStudyId in number(10)

vSiteCode in varchar2(10)

nSiteId in number(10)

vTestProd in varchar2(1)

vStudySite in varchar2(10)

vReturnMsg out varchar2(60)

Check That a Study Site Code Can Be Changed

Checks whether a STUDY_SITE code can be changed.

LocalStudySite.ChangeStudySite
Param = vStudyCode in varchar2(10)
nStudyId in number(10)

vSiteCode in varchar2(10)

nSiteId in number(10)

vTestProd in varchar2 1)

vOldStudySite in varchar2(10)

vNewStudySite in varchar2(10)

vReturnMsg out varchar2(60)

Table Details

The following tables show the structure of the interface tables for Oracle Clinical, grouped by subsystem. The columns indicated as NOT NULL are mandatory for Oracle Clinical functionality. The other columns may not be used at all, or may be used in a display-only report or screen.

Drug Supply System

There are two drug supply tables, OCL_DOSAGE_FORMS and OCL_PRODUCT_MASTERS.

OCL_DOSAGE_FORMS

This is a list of the forms in which drugs can be manufactured. It adds a description to the dosage form code held on OCL_PRODUCT_MASTERS.

Parameter	Datatype, Null?	Description
DOFO_ID	NUMBER(10), NOT NULL	Unique ID for the dosage form
DOSAGE_FORM_CODE	VARCHAR2(7), NOT NULL	User-understood code for the dosage form

OCL_PRODUCT_MASTERS

This is a list of two types of material. The ACT SUB (active substances) materials are base chemicals that can be assigned to programs. The FORMULATED materials are formulated drug supplies (for example, 250 mg pill of a drug) from which treatment regimens can be defined.

Parameter	Datatype, Null?	Description
PM_ID	NUMBER(10), NOT NULL	Unique ID for the product master
PRODUCT_NAME	VARCHAR2(70), NOT NULL	User-understood name for the product master
PM_STATUS_CODE	VARCHAR2(15), NOT NULL	Set to ACTIVE
PM_TYPE_CODE	VARCHAR2(10), NOT NULL	ACTIVE SUB—raw compounds to which programs can be linked FORMULATED—formulated drugs used in treatment regimens
CREATION_TS	DATE, NOT NULL	Date and time the record was created
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
ACTIVE_SUBSTANCE_PM_ID	NUMBER(10)	Product Master ID of the active substance in a formulated product (Null for active substances themselves)
DOFO_ID	NUMBER(10)	ID of the dosage form for formulated products (Null for active substances)
CODE	VARCHAR2(10)	Code for the active substance itself or for the active substance in a formulated product
SUBCODE	VARCHAR2(10)	Subcode for the active substance itself or for the active substance in a formulated product
MODIFICATION_TS	DATE	Date and time the record was last modified
MODIFIED_BY	VARCHAR2(30)	User ID of the person who last modified the record

Investigator Management System

There are four investigator management system tables, OCL_INVESTIGATORS, OCL_SITES, OCL_STUDY_SITES, and OCL_STUDY_SITE_ROLES.

OCL_INVESTIGATORS

This is a master list of investigators who can be assigned to a study.

Parameter	Datatype, Null?	Description
INVESTIGATOR_ID	NUMBER(10,0), NOT NULL	ID of an investigator record

Parameter	Datatype, Null?	Description
INVESTIGATOR	VARCHAR2(10),NOT NULL	Code for the investigator as recognized by the user
TITLE	VARCHAR2(4), NULL	Title for the investigator
INITIALS	VARCHAR2(4), NULL	Initials of the investigator
FIRST_NAME	VARCHAR2(15),NOT NULL	First name of the investigator
LAST_NAME	VARCHAR2(20),NOT NULL	Last name of the investigator
ACTIVE_FLAG	VARCHAR2(1), NOT NULL	Y = Investigator is available to work on new studies N = Investigator not assignable to new studies
CREATED_BY	VARCHAR2(30),NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
PHONE_NUMBER	VARCHAR2(25),NOT NULL	Phone number of the investigator
COUNTRY	VARCHAR2(7), NULL	Region code of the country in which the investigator lives
STATE	VARCHAR2(7), NULL	Region code of the state/province/county, etc. where the investigator lives
CITY	VARCHAR2(25),NULL	Name of the city where the investigator lives
ADDRESS_NAME	VARCHAR2(40),NULL	Name to start the investigator's address
ADDRESS_LINE_1	VARCHAR2(40),NULL	First line of the investigator's address
ADDRESS_LINE_2	VARCHAR2(40), NULL	Second line of the investigator's address
ADDRESS_LINE_3	VARCHAR2(40), NULL	Third line of the investigator's address
POSTAL_CODE	VARCHAR2(15),NULL	Postal or ZIP code for the investigator
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_SITES

This is a list of sites where a study can be performed.

Parameter	Datatype, Null?	Description
SITE_ID	NUMBER(10,0), NOT NULL	ID for the site
SITE	VARCHAR2(10), NOT NULL	Code for the site as recognized by the user
ACTIVE_FLAG	VARCHAR2(1), NOT NULL	Y = Site is available for work on new studies N = Site not assignable to new studies
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
PHONE_NUMBER	VARCHAR2(25), NOT NULL	Phone number for the site
NAME	VARCHAR2(60), NOT NULL,	Name to start the site's address
COUNTRY	VARCHAR2(7), NOT NULL	Region code of the country where the site is located

Parameter	Datatype, Null?	Description
STATE	VARCHAR2(7), NOT NULL	Region code of the state/province/county, etc. where the site is located
CITY	VARCHAR2(25), NOT NULL	City where the site is located
ADDRESS_LINE_1	VARCHAR2(40), NOT NULL	First line of the site's address
ADDRESS_LINE_2	VARCHAR2(40), NULL	Second line of the site's address
ADDRESS_LINE_3	VARCHAR2(40), NULL	Third line of the site's address
POSTAL_CODE	VARCHAR2(15), NOT NULL	Postal or ZIP code of the site
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record

OCL_STUDY_SITES

This is a record of the assignments of a site to a study. The Oracle model assigns sites to a study and then patients to a site for the use of the investigator currently responsible for managing the site. The OCL_STUDY_SITES table is the linchpin around which the assignment of patients and investigators is maintained.

Parameter	Datatype, Null?	Description
SITE_ID	NUMBER(10,0), NOT NULL	ID of the site assigned to the study
CLINICAL_STUDY_ID	NUMBER(10,0), NOT NULL	ID of the study to which the site has been assigned
DATA_FREEZE_FLAG	VARCHAR2(1), NOT NULL	Y = All data for the site is frozen N = Data for site can be changed
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
OK_TO_SHIP_FLAG	VARCHAR2(1), NOT NULL	Y = Site is approved to receive drug supplies N = Site is awaiting approval
START_DATE	DATE, NULL	Date the site was first assigned to the study
END_DATE	DATE, NULL	Date the site was closed
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record.
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_STUDY_SITE_ROLES

This is a record of the assignment of an investigator to a site in a study. Only one investigator can be responsible for a site within a study at a time.

Parameter	Datatype, Null?	Description
CLINICAL_STUDY_ID	NUMBER(10,0), NOT NULL	ID of the study to which the site has been assigned
SITE_ID	NUMBER(10,0), NOT NULL	ID of the site assigned to the study
INVESTIGATOR_ID	NUMBER(10,0), NOT NULL	ID of the investigator assigned to the site
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
CURRENT_FLAG	VARCHAR2(1), NOT NULL	Y = This is the investigator currently responsible for the site N = This is a historic assignment
START_DATE	DATE, NOT NULL	Date the assignment started
NUMBER_OF_ PATIENTS_REQUIRED	NUMBER(10,0), NULL	Number of patients the investigator is targeted with recruiting at this site
CONTRACT_DATE	DATE, NULL	Date the investigator's contract was signed
EXPECTED_ ENROLLMENT_RATE	NUMBER(5,2), NULL	Number of patients the investigator is expected to recruit each month
TERMINATION_DATE	DATE, NULL	Date the investigator stopped being responsible for the site.
DISCONTINUATION_ LETTER_DATE	DATE, NULL	Date the investigator's termination letter was sent
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record

Study Planning

There are six study planning tables, OCL_ORGANIZATION_UNITS, OCL_ PROGRAMS, OCL_PROJECTS, OCL_PROGRAM_PRODUCT_MASTERS, OCL_ STUDIES, and OCL_STUDY_REGIONS.

OCL_ORGANIZATION_UNITS

This table contains a list of organizations within the company that manage or sponsor studies. A study must be assigned to one of these organizations.

Parameter	Datatype, Null?	Description
ORGANIZATION_ UNIT_ID	NUMBER(10), NOT NULL	Unique ID for the organization unit
CODE	VARCHAR2(3), NOT NULL	Code for the organization unit as recognized by the user

Parameter	Datatype, Null?	Description
NAME	VARCHAR2(60), NULL	Name for the organization unit
DESCRIPTION	VARCHAR2(70), NOT NULL	Description of the organization unit
START_DATE	DATE, NOT NULL	Date the organization unit became active
END_DATE	DATE, NULL	Date the organization unit was retired
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record.
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified.

OCL_PROGRAMS

Programs are the upper tier in the studies hierarchy. They group the studies within a particular compound, which could include multiple indications and formulations of the compound across all locations where studies are being performed.

Parameter	Datatype, Null?	Description
PROGRAM_CODE	VARCHAR2(15), NOT NULL	User code for the program
DESCRIPTION	VARCHAR2(70), NOT NULL	Description of the program
ACTIVE_FLAG	VARCHAR2(1), NOT NULL	Y = Program is still active N = Program is for historical use only
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_PROJECTS

Projects are the lower tier in the studies hierarchy. They group studies in a program into some customer-determined groups. Typical groups could split studies by indications or formulations of the compound, and possibly by locations or regions.

Parameter	Datatype, Null?	Description
PROJECT_CODE	VARCHAR2(15), NOT NULL	Code for the project
DESCRIPTION	VARCHAR2(70), NOT NULL	Description of the project
START_DATE	DATE, NOT NULL	Date the project started
END_DATE	DATE, NULL	Date the project ended
PROGRAM_CODE	VARCHAR2(15), NOT NULL	Code of the program to which the project belongs

Parameter	Datatype, Null?	Description
CREATED_BY	VARCHAR2(30), NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_PROGRAM_PRODUCT_MASTERS

This lists type ACT SUB (active substance) product masters, assigned to programs.

Parameter	Datatype, Null?	Description
PROGRAM_CODE	VARCHAR2(15), NOT NULL	Code of the program concerned
PM_ID	NUMBER(10,0), NOT NULL	ID of the product master assigned
PRIMARY_RS_ COMPOUND_FLAG	VARCHAR2(1), NOT NULL	Y = Main compound of the program N = Other compounds assigned to the program
CREATED_BY	VARCHAR2(30),NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_STUDIES

This table lists all studies within a set of instances that share data — all studies ever performed, currently being performed, or might be performed in the future. A study must exist in this table before it can be designed in Oracle Clinical.

Parameter	Datatype, Null?	Description
TASK_ID	NUMBER(10), NOT NULL	Unique ID for the planned study
STUDY	VARCHAR2(15),NOT NULL	User-recognized code of the study
CREATED_BY	VARCHAR2(30),NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
ORGANIZATION_ UNIT_ID	NUMBER(10), NOT NULL,	Code of the organization sponsoring the study
PROJECT_CODE	VARCHAR2(15),NOT NULL	Code of the project to which the study belongs

Parameter	Datatype, Null?	Description
PROGRAM_CODE	VARCHAR2(15),NOT NULL	Code of the program that owns the project to which the study belongs
TITLE	VARCHAR2(255), NULL	Title for the study
INVESTIGATORS_ PLANNED	NUMBER(10), NULL	Planned number of investigators in the study
EXPTL_DESIGN_ TYPE_CODE	VARCHAR2(5), NULL	Type of design being used in the study
REGION_ID	NUMBER(10), NULL	ID of the master region for the study
CLINICAL_PHASE	VARCHAR2(5), NULL	Clinical phase of the study
MODIFIED_BY	VARCHAR2(30),NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

OCL_STUDY_REGIONS

This table lists the regions where a study is to be performed. One region must be flagged as the master region— the one displayed in the heads of some reports.

Parameter	Datatype, Null?	Description
STUDY_REGION_ID	NUMBER(10), NOT NULL,	ID for the study region record
TASK_ID	NUMBER(10), NOT NULL	ID of the study concerned
REGION_ID	NUMBER(10), NOT NULL,	ID of the region concerned
REGION_CODE	VARCHAR2(7), NOT NULL	Code of the region concerned
REGULATORY_ FLAG	VARCHAR2(1), NOT NULL	Y = Region is being used for regulatory reasons N = Region is not being used for regulatory reasons
MARKETING_FLAG	VARCHAR2(1), NOT NULL	Y = Region is being used for marketing reasons N = Region is not being used for marketing reasons
REPORTING_ REGION_FLAG	VARCHAR2(1), NOT NULL	Y = Region is the main reporting region N = Region is not the main reporting region
CREATED_BY	VARCHAR2(30),NOT NULL	User ID of the person who created the record
CREATION_TS	DATE, NOT NULL	Date and time the record was created
MODIFIED_BY	VARCHAR2(30), NULL	User ID of the person who last modified the record
MODIFICATION_TS	DATE, NULL	Date and time the record was last modified

4

Web Services

This section includes the following topics:

Oracle Study, Subject, and Visit Synchronization Process Integration Pack (PIP) for Siebel Clinical and Oracle Clinical introduces functionality to support the integration between the participating applications. Oracle Clinical provides study, subject, and visit functionalities as inbound web services which create object metadata in Oracle Clinical using information from the other system.

This chapter provides instructions on how to configure message-level security for Oracle Clinical web services to secure some or all of the message content. This chapter, however, does not cover the many options and alternatives to implement web services security (WS-Security, or WSS), or applying transport-level security (TLS).

> **Note:** By default, Oracle Clinical web services are enforced with **oracle/wss11_x509_token_with_message_protection_service_policy** to ensure message-level protection and certificate-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard. Make sure to configure the **oracle/wss11_x509_token_with_message_protection_client_policy** policy on the AIA server to provide message protection and certificate credential population for outbound SOAP requests.

Setting Up Web Services

This section contains the following topics:

You must apply the Oracle Clinical web services to a computer running Oracle Fusion Middleware, which can be the Oracle Clinical Middle-tier server, the SOA Suite server, or any other server running Oracle Fusion Middleware.

The Oracle Clinical web services are installed in OPA_HOME\webservices\X\OracleClinical.ear, where the X represents the currently installed version. For example, 11g.

Prerequisites

Before you begin to set up the web services, you should check that you meet the following requirements:

- Have administrative access to the AIA server so that you can perform setup tasks in the SOA Home
- Have administrative access to the Oracle Clinical or RDC web server
- Have an X.509 and root certificate from a trusted certificate authority

> **Note:** The steps in this chapter generate a self-validating certificate, not a trusted certificate. Oracle does not recommend that you secure your system with generated, self-signed certificates, but instead recommends that you obtain an X.509 and root certificate from a trusted certificate authority. Allow some time for the certificate to be issued.

- Have downloaded the Oracle Repository Creation Utility (RCU)
- Have downloaded OpenSSL for Windows.

Installing the Oracle Clinical Web Services

You should apply the Oracle Clinical web services to a computer running Oracle Fusion Middleware. Alternately, you can use the Oracle Clinical Middle-tier server for this purpose.

If you are upgrading to Oracle Clinical Release 5.1 web service usage, you must set up the web service client with the oracle/wss11_x509_token_with_message_protection_client_policy assertion to sign messages before the Study, Subject, and Visit Synch for the Siebel Clinical - Oracle Clinical PIP can call the Oracle Clinical web services. To enforce this policy for the client, follow the instructions in step 5 of Client Side Configuration

To install the Oracle Clinical web services, follow the high-levels steps below, in the given order.

1. Run the Oracle Repository Creation Utility (RCU) to create a metadata repository. Oracle Web Service Manager (OWSM) uses this repository to secure policies.

   RCU requires access to a database and you can use any compatible database.

2. Install the Oracle Web Service Manager (OWSM) on the Oracle RDC Middle-tier web server.

   See "Installing the Oracle Web Service Manager (OWSM)" on page 4-3 for more information on how to complete this step.

3. Install the Oracle Clinical web services in the WebLogic server.

   See "Installing the Oracle Clinical Web Services in the WebLogic Server" on page 4-4 for more information on how to complete this step.

4. Configure the certificates between the Oracle Clinical web service server and the AIA server.

   See "Configuring Certificates" on page 4-6 for more information on how to complete this step.

You need to request a certificate from a Certificate Authority (CA) vendor such as Thawte, Entrust, or Verisign. Once you have obtained the X.509 certificate, install it for use by the Oracle Clinical web server. Allow some time for the certificate to be issued.

5. Create the keystore and credential store on the Oracle Clinical web service server.
6. Add the WSDL definition information to the AIA system.

   See "Client Side Configuration" on page 4-10 for more information on how to complete this step.

Installing the Oracle Web Service Manager (OWSM)

The Oracle Web Service Manager (OWSM) applies server policies on the Oracle Clinical web service application. OWSM requires access to a database and you can use any compatible database.

You can install OWSM from the Quick Start installation program that installs the Oracle Enterprise Manager.

> **Important:** The Oracle Repository Creation Utility (RCU) must be run prior to installing the Oracle Web Service Manager (OWSM).

To install OWSM:

1. Click on the Windows **Start** button and select **All Programs**.
2. Find and click on the **Oracle WebLogic** folder and select **QuickStart**.

   The WebLogic Platform Quick Start screen opens.
3. Click on **Getting started with WebLogic Server 10.3.6**.

   The Fusion Middleware Configuration Wizard opens to the Welcome page.
4. Select **Create a new WebLogic domain** and click **Next**.

   The Select Domain Source page opens.

   > **Note:** If you already have a domain, you can extend the existing domain to install OWSM. To do so, for this step select the **Extend an existing WebLogic domain** checkbox and proceed to the next step.

5. Select **Generate a domain configured automatically to support the following products** and then select the checkboxes for **Oracle Enterprise Manager - 11.1.1.0 [oracle_common]** and **Oracle WSM Policy Manager - 11.1.1.0 [oracle_common]**. **Oracle JRF - 11.1.1.0 [oracle_common]** is automatically selected, leave it as is.

   Click **Next**.

   The Specify Domain Name and Location page opens.
6. Replace the default values with your domain name and location.

   > **Note:** Save the entered domain name and location. This information is required in later steps.

   Click **Next**.

The Configure Administrator User Name and Password page opens.

7. Enter your username and password, and then click **Next**.

   The Configure Server Start Mode and JDK page opens.

8. Select **Production Mode** in the left pane and **Available JDKs** in the right pane.

   Click **Next**.

   The Configure JDBC Component Schema page opens.

9. Fill in the fields with your information. This information should match the information used to run the RCU.

   The Driver information is set by default. Leave it at the default setting.

   Click **Next**.

   The JDBC Component Schema page opens. This page confirms that the information on the previous page is correct to use when installing the OWSM Policy Manager.

10. Click **Next**.

    The Select Optional Configuration page opens.

11. If you are installing in a WebLogic server that was installed for Oracle Clinical RDC, select the **Administration Server** and **Managed Servers, Clusters and Machines** checkboxes to change the default port numbers for the servers.

    Otherwise, click **Next** on the following screens without making any changes until you reach the Configuration Summary page, then continue with step 15.

12. Click **Next**.

    The Configure the Administration Server page opens.

13. Change the default listen port and then click **Next**.

    Note: The port you use must be different from the ones used for any other running servers.

    The Configure Managed Servers page opens.

14. Change the default listen port and then click **Next**.

    The Configuration Summary page opens.

15. Verify the details of the domain and components that you selected for installation, and then click **Create** to continue the installation.

    The Creating Domain page opens.

16. When progress reaches 100%, click **Done** to complete the installation and exit the wizard.

You can access the admin server console for the new domain at http://*middletier_name*:*admin_server_port*/console.

Installing the Oracle Clinical Web Services in the WebLogic Server

1. Log into the WebLogic server console as a WebLogic user.
2. In the Domain Configurations panel, under the Deployed Resources section, select **Deployments**.

The system displays all of the applications currently deployed on this server.

3. Click **Install** to deploy all of the options.
4. Click the **Upload your file(s)** option to upload the .ear file from your local machine.
5. Click **Next**.
6. Click **Next** and then choose the option **Install this deployment as an application**.
7. Click **Next**.
8. Choose from the available targets, such as soa_server1, and click **Next**.
9. Set `OracleClinical-WS` as the name of the application. Then click **Next**.

> **Note:** Do not change any of the other default values.

The system displays your configuration details. You can edit the details by navigating back through the screens.

10. Click **Finish**.

To confirm that the web services deployed successfully:

1. Go to the list of Deployments and verify that the **OC Webservices** project is in the Active state.

   If it is in the New state, start the managed server from **Environment** > **Servers**.
2. Click on the application and select the **Testing** tab. There should be three Test Points available to check the three web services: InvestigatorService, Site Service, and StudySite Service.

   Before you can test these web services, you must create a data source. See To create a data source: below for instructions on how to do this. You can create the data source in the current session.
3. Once the data source is created, select the link **Test Client** under Test Point for each web service to check that it is working properly.

To create a data source:

1. In the same console screen from the previous section, under Domain Structure, click **Services**, and then click **Data Sources**.

   The system displays all existing data sources.
2. Click **New**.

   Three options are available.
3. Select the **Generic Data Source** option.
4. In the Create Screen, name the datasource `OracleClinicalCoreDS`.
5. In the JNDI Name column, enter `jdbc/OracleClinicalCoreDS`.

> **Note:** Do not change any other settings from their default values.

6. Click **Next**.
7. Click **Next**.

8. Click **Next** and enter the database name, host name, and port of your Oracle Clinical database. You must enter `rxa_ws` as the username and the password for the rxa_ws user.
9. Click **Next**.
10. Select your SOA server as the target. For example, select soa_server1.
11. Click **Finish**. The data source is created and deployed on the target server.
12. Verify that the data source was successfully created in the list of data sources. Click on the data source and go to the **Monitoring** tab.
13. Select the **Testing** tab and test the data source.

Configuring Certificates

Configuring the certificates between the Oracle Clinical web service server and the AIA SOA server involves populating keystores and generating public certificates that you copy between the servers.

In the WebLogic server, a keystore is used to store credentials, such as certificates. The Credential Store Framework (CSF) provides a way to store, retrieve, and delete credentials for a web service and other applications. OWSM uses the CSF to manage the credentials in a secure form, for example, by retrieving alias names and passwords for keys in the Java keystore or usernames and passwords used for authentication.

Keytool is a utility that Oracle delivers with Java. The system uses this utility to manage the import and export of certificates from the keystores. You must have access to the Keytool utility to continue with the installation. If it isn't already, you need to add it to your environment variable path, the JDK path up to bin.

When using WSS11 policies, configuration is required on both sides, as follows:

- On the service side, where the Oracle Clinical web service is deployed, set up private keys and define them as the encryption key alias in the OWSM Keystore Configuration screen.
- On the client side, the AIA server, you need to configure the client-side trust by obtaining the server's certificate.

Obtaining a CA Certificate

You need to request a certificate from a Certificate Authority (CA) vendor such as Thawte, Entrust, or Verisign, then include it in the keystore.

To get the certificate, you must create a certificate request and submit it to the CA. The CA authenticates the certificate requestor and creates a digital certificate based on the request. Allow some time for the certificate to be issued.

> **Note:** If you do not have an X.509 certificate, you can generate a certificate using the Keytool utility, but Oracle does not recommend that you secure your system with generated self-signed certificates.

Sample Configuration

The Server Side Configuration and Client Side Configuration sections that follow serve as an example of how to set up and implement AIA to call an Oracle Clinical web service in a different domain. The process also involves securing the AIA client and web service using a policy that is created based on the **oracle/wss11_x509_token_**

with_message_protection_service_policy, and changing its configuration to only sign the header and not the encryption.

Server Side Configuration

In this sample configuration we are using the OpenSSL certificate management tool to demonstrate the process.

1. Download OpenSSL for Windows from https://code.google.com/p/openssl-for-windows/downloads/list
2. Install the application in C:\OpenSSL and place the executables in the C:\OpenSSL\bin folder instead of the Windows system folder.

   You can install the application in another folder, but the above path is recommended.
3. Once the installation is complete, add the new bin folder to your system path as follows:
   a. Right-click on **My Computer** and select **Properties**.
   b. Go to the **Advanced** tab and click **Environment Variables**.
   c. Under System Variables, select **Path** and click **Edit**.
   d. If the Path value does not end with a semicolon, add one.
   e. Add a path to the bin folder. For example, C:\OpenSSL\bin.
   f. Click **OK** three times to finish.
4. Generate the certificate authority as shown in the steps below, which are performed in the Command Prompt. In the example, the alias *webservice* is used.
   a. Create directories in which to store the certificates using the below commands:

      ```
      mkdir client
      mkdir server
      mkdir ca
      ```

   b. Generate an RSA private key for the certificate, called *ca.key*.

      ```
      Openssl genrsa -des3 -out ca/ca.key 4096
      ```

      When the message `Enter pass phrase for ca/ca.key:` is displayed, type in a password. For example, *welcome*.
   c. Create the master certificate, called *ca.cer*, using the key generated before.

      ```
      openssl req -new -x509 -days 365 -key ca/ca.key -out ca/ca.cer -subj
      "/CN=HSGBUSupportCA/OU=HSGBU/O=Oracle/L=Burlington/ST=Massachusetts/C=US"
      -config OpenSSLpath\openssl.cnf
      ```

      OpenSSLpath stands for the path for the directory where OpenSSL was installed at step 2.

      When the message `Enter pass phrase for ca/ca.key:` is displayed, type in the password used for the above step.
5. Generate server certificates by following the steps below:
   a. Use the keytool command -genkeypair to generate the key pair for the alias *servercer*. The key pair is used for SSL configuration on the server.

      ```
      keytool -genkey -keyalg RSA -alias servercer -keypass welcome -keystore
      server/server_keystore.jks -storepass welcome -dname
      ```

```
"CN=<servername>,OU=HSGBU,O=Oracle,L=Burlington,ST=Massachusetts,C=US"
-validity 365
```

b. Generate the certificate request for *servercer* using the -certreq command. This generates a certificate request for the *servercer* alias and a certificate signing request (CSR) named *server/servercer.csr*.

```
keytool -certreq -alias servercer -sigalg "SHA1withRSA" -file
server/servercer.csr -storetype jks -keystore server/server_keystore.jks
-storepass welcome
```

c. Sign the CSR file using the CA created above. Run the command:

```
keytool -list -v -alias servercer -keystore server/server_keystore.jks
-storepass welcome
```

The response for the above command also returns the certificate serial number, which is needed for the following step. You can find it in the line starting with *Serial Number*.

d. Submit the CSR file to a CA, which authenticates the request and returns a certificate or certificate chain.

```
openssl x509 -req -days 365 -in server/servercer.csr -CA ca/ca.cer -CAkey
ca/ca.key -set_serial 0x<serial number from above step> -out
server/servercerIssuedByCA.cer
```

e. Import the CA root certificate, which authenticates the CA's public key.

```
keytool -importcert -alias 'oracleHsgbuca' -trustcacerts -file ca/ca.cer
-keystore server/server_keystore.jks -storepass welcome
keytool -list -keystore server/server_keystore.jks -storepass welcome
```

f. Replace the self-signed certificate with the trusted CA certificate issued by the CA in response to the certificate request using the -importcert keytool command. This replaces the self-signed certificate *servercer* with the trusted CA certificate named *server/servercerIssuedByCA.cer*.

```
keytool -importcert -trustcacerts -alias servercer -file
server/servercerIssuedByCA.cer -keystore server/server_keystore.jks
-storepass welcome
keytool -list -keystore server/server_keystore.jks -storepass welcome
```

The *HSGBUSupportCA* (trustedCertEntry) and the *servercer* (PrivateKeyEntry) certificates are stored in the **server/server_keystore.jks**. The server/server_ keystore.jks file is used to configure web services for the WebLogic server.

6. Generate certificates for the client, the AIA server, by following these steps:

a. Use the keytool command -genkeypair to generate the key pair for the alias *webservice*, which is later used for encryption.

```
keytool -genkey -keyalg RSA -alias webservice -keypass welcome -keystore
client/client_keystore.jks -storepass welcome -dname "CN=webservice"
-validity 365
```

b. Use the -importcert keytool command to import the trusted CA root certificate, named *ca/ca.cer* here, using the alias *ocalehsgbuca*, into the *client-keystore.jks* keystore.

```
keytool -importcert -alias "oracleHsgbuca" -trustcacerts -file ca/ca.cer
-keystore client/client_keystore.jks -storepass welcome
```

c. Generate the certificate request using the -certreq keytool command. The below command generates a certificate request for the *webservice* alias and a CSR named *client/webservice.csr*.

```
keytool -certreq -alias webservice -sigalg "SHA1withRSA" -file
client/webservice.csr -storetype jks -keystore client/client_keystore.jks
-storepass welcome
```

d. Get the serial number for the certificate, which is needed in the following step.

```
keytool -list -v -alias webservice -keystore client/client_keystore.jks
-storepass welcome
```

e. Submit the CSR file to a CA for authentication. The CA returns a certificate or certificate chain.

```
openssl x509 -req -days 365 -in client/webservice.csr -CA ca/ca.cer -CAkey
ca/ca.key -set_serial 0x<serial number from above step> -out
client/webserviceIssuedByCA.cer
```

f. Use the -importcert keytool command to replace the self-signed certificate with the trusted CA certificate issued by the CA in response to the certificate request. The following command replaces the self-signed certificate for the alias *webservice* with the trusted CA certificate named *client/webserviceIssuedByCA.cer*.

```
keytool -importcert -trustcacerts -alias webservice -file
client/webserviceIssuedByCA.cer -keystore client/client_keystore.jks
-storepass welcome
```

g. List the content of the keystore *client/client_keystore.jks*.

```
keytool -list -keystore client/client_keystore.jks -storepass welcome
```

h. Import the public server certificate for the encryption.

```
keytool -importcert -trustcacerts -alias servercer -file
server/servercerIssuedByCA.cer -keystore client/client_keystore.jks
-storepass welcome
```

i. List the content of the keystore *client/client_keystore.jks* and *server_keystore.jks*.

```
Keytool -list -keystore client/client_keystore.jks -storepass welcome
Keytool -list -keystore server_keystore.jks -storepass welcome
```

7. Configure the security provider on Enterprise Manager for the OCWebService domain by following the below steps.

a. Log into the Fusion Middleware control for the domain at http*servername*:*port*)\em.

b. Right-click on the domain you are configuring and select **Security**, then **Security Provider Configuration**.

c. In the Keystore section, click **Configure**.

d. Fill in the fields with the below information:

Keystore Type: JKS
Keystore Path: C:/x509/server/server_keystore.jks (server keystore location used for step 4-a.)
Keystore Password: welcome (used in the commands to create the keystore)
Signature Key Alias: servercer (created using the keytool command)

Alias Password: welcome (created using the keytool command)
Encryption Key Alias: servercer (created using the keytool command)
Alias Password: welcome (created using the keytool command)

e. Click **OK**.

8. Configure the credentials for the web service server as shown below.

   a. Right-click on the domain you are configuring and select **Security**, then **Credentials**.

   b. Make sure there is an entry under the Credentials tree with the name *oracle.owsm.security*.

      The content of these entries must be based on the information used in the previous step. The same passwords apply.

   c. Add a new realm from the administration console at http*servername*:*port*)\console.

      To do so, under Domain, click on **Security Realms** and select the default realm. Select the **Users and Groups** tab and click **New**. Provide the required information, making sure that the name is the same one that was used to generate the certificate on the client side. Click **OK**.

   d. Once the configuration is complete:

   - Restart all the affected managed servers for the new configurations to take effect.
   - Restart the admin server.

Client Side Configuration

Configuring the client side can be performed by uploading a new WSDL file to the SOA server. In the AIA layer, the Enterprise Business Service (EBS) calls the Oracle Clinical provider using routing rules defined in the EBS. Web Services Description Language (WSDL) requires web services to communicate between Oracle Clinical and the EBS. For details on how to configure EBS and web services, see the *Oracle Study, Subject, and Visit Synchronization Integration Pack for Siebel Clinical and Oracle Clinical Implementation Guide*, available at http://docs.oracle.com/cd/E51618_01/doc.111/e36156/toc.htm.

After you complete the Oracle Clinical web service installation process, verify that the security policy is configured for web services and save the WSDL files in order to upload them to the client. The Oracle Clinical Web Services Description Language (WSDL) files can be located by accessing the deployed Oracle Clinical web service through any web browser. You can display the WSDL of the web service in your browser to ensure that it has deployed correctly using the following URL:

http://*host*:*port*/*contextPath*/*serviceUri*?WSDL

Where:

host refers to the computer on which the Oracle Clinical web service, deployed on the WebLogic server is running.
port refers to the port number on which the WebLogic Server is listening.
contextPath refers to the context root of the web service.
serviceUri refers to the value of the serviceUri attribute, such as InvestigatorService, SiteService, or StudySiteService.

Note: For an Oracle Clinical web service associated with an **oracle/wss11_x509_token_with_message_protection_service_policy** policy file, published as an attachment to the WSDL by default, the static WSDL file in the web service archive file (JAR or WAR) is different from the dynamic WSDL of the deployed web service because only the latter includes specific <SecurityToken> elements.

To configure the client side:

1. Navigate to $AIA_HOME/AIAMetaData/AIAComponents/ApplicationObjectLibrary/OracleClinical/V1/wsdls.
2. Take the backup of **InvestigatorService.wsdl**, **SiteService.wsdl**, and **StudySiteService.wsdl**.
3. Copy the updated WSDL file.
4. Upload the new WSDL file to the MDS following the instructions below.
   a. Navigate to the $AIA_INSTANCE/config directory.
   b. Create a backup of **UpdateMetaDataDp.xml**.
   c. Edit **UpdateMetaDataDp.xml** file as follows:

```
<?xml version="1.0" standalone="yes"?>
<DeploymentPlan component="Metadata" version="3.0">
  <Configurations>
    <UpdateMetadata wlserver="fp" >
      <fileset dir="${AIA_HOME}/AIAMetaData">
        <include
name="AIAComponents/ApplicationObjectLibrary/OracleClinical/V1/wsdls/*.wsdl
"/>
      </fileset>
    </UpdateMetadata>
  </Configurations>
</DeploymentPlan>
```

   d. Navigate to the $AIA_INSTANCE/bin directory.
   e. Execute one of the following commands:

      For Windows, execute *aiaenv.bat*.

```
ant -f %AIA_HOME%\Infrastructure\Install\config\UpdateMetaData.xml
```

      For Linux, source *aiaenv.sh*.

```
ant -f $AIA_HOME/Infrastructure/Install/config/UpdateMetaData.xml
```

   f. Once the build completes, restart the SOA server.

Important: The SOA server should be restarted every time a new WSDL file is uploaded.

5. Attach the oracle/wss11_x509_token_with_message_protection_client_policy policy to the client side.
   a. Log into the SOA control panel for the domain at http*SOAservername*:*port*\em.

b. Under SOA, right-click on **UpdateClinicalStudyOCHealthSciencesProvABCSImpl** and select **Service/References Properties**, then select the name of the web service.

c. Go to the **Policies** tab and then click **Attach/Detach**.

A new window opens.

d. In the Available Policies pane, find and select the client policy, then click **Attach**.

e. Repeat the above steps as necessary to attach the policy to other services.

6. Configure the security provider on Enterprise Manager for the client.

a. Log into the SOA control for the domain.

b. Right-click on the domain you want to configure and select **Security**, then **Security Provider Configuration**.

c. In the Keystore section, click **Configure**.

d. Provide the information generated in the first section of the document.

Keystore Type: JKS
Keystore Path: /slot/ems6679/oracle/Keystore/x509_latest/client/client_keystore.jks (path for the client keystore, used for step 6 of Server Side Configuration)
Keystore Password: welcome (used in the commands to create the keystore)
Signature Key Alias: webservice (created using the keytool command)
Alias Password: welcome (created using the keytool command)
Encryption Key Alias: webservice (created using the keytool command)
Alias Password: welcome (created using the keytool command)

e. Click **OK**.

f. On the client machine, navigate to the directory /slot/ems6679/oracle/Middleware/user_projects/domains/soa_domain/bin/.

g. Open the **startWebLogic.sh** file.

h. Change the line `SAVE_JAVA_OPTIONS ="${JAVA_OPTIONS}"` into `SAVE_JAVA_OPTIONS="${JAVA_OPTIONS} -Djavax.net.ssl.trustStore=`*`full path to keystore`*`"`.

7. Configure the credentials for the client server.

a. Right-click on the domain you are configuring and select **Security**, then **Credentials**.

b. Check that there is an entry called *oracle.owsm.security* under the Credentials tree.

If the entry doesn't exist, redo step 6.

8. Add a new realm on the client server.

a. Go to the administration console at http*servername*:*port*\console.

b. Select **Security Realms** and then click on the default realm.

c. Select the **Users and Groups** tab and then click on **New** in the Users pane.

The Create a New User page opens.

d. Set up the details for the new user.

Note: Provide the same alias as used to generate the client side certificate. In this case, *webservice*.

e. Click **OK**.

Available Web Services

The following Web services are available:

- "Investigator Service" on page 4-13
- "Site Service" on page 4-16
- "Study Site Service" on page 4-18
- "Partial Source Data Verification Web Service" on page 4-23

Web services are available in Oracle Clinical to automatically create and update Sites, Investigators, Study Sites, and Investigator assignments from information passed from an external system. These services were developed for use in the Oracle Clinical integration with Siebel Clinical, but they are available for use in integrating Oracle Clinical with other systems as well.

Investigator Service

The Investigator service allows you to create, update, request and delete Investigators.

InvestigatorService has the following methods:

- "Create Investigator" on page 4-13
- "Update Investigator" on page 4-14
- "Request Investigator" on page 4-15
- "Delete Investigator" on page 4-15

Create Investigator

This method creates a new Investigator in Oracle Clinical.

Web Service Name: InvestigatorService

Method Name: CreateInvestigator

Return Type: Investigator object set to Active and with INVESTIGATOR_ID and other autopopulated values populated.

Arguments: The following table displays the method's arguments.

Table 4–1 CreateInvestigator Method Arguments

Argument Name	Type	Description
Investigator	String	INVESTIGATOR_ID; the unique numeric database code for the Investigator. If the code is greater than ten characters, the system uses the generated Investigator ID instead.
Title	String	The Investigator's title; for example, `Dr`.
First Name	String	The Investigator's given name.
Initials	String	The Investigator's initials.

Table 4–1 (Cont.) CreateInvestigator Method Arguments

Argument Name	Type	Description
Last Name	String	The Investigator's family name.
Phone Number	String	The Investigator's telephone number.
Address Name	String	The Investigator's institution.
AddressLine1	String	The first line of the address of the Investigator.
AddressLine2	String	The second line of the address of the Investigator, if any.
AddressLine3	String	The third line of the address of the Investigator, if any.
City	String	The city in the Investigator's address.
State	String	The state must be a valid region code defined in the Regions form in Oracle Clinical.
Country	String	The country must be a valid region code defined in the Regions form in Oracle Clinical.
Postal Code	String	The postal code in the Investigator's address.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Update Investigator

This method updates an existing Investigator in Oracle Clinical. If no current Investigator matches, the method returns an error. If you pass the Oracle Clinical Investigator ID, Oracle Clinical uses that to identify the correct Investigator. If not, the system uses the Investigator code, which may have originated in the source system.

Web Service Name: InvestigatorService

Method Name: UpdateInvestigator

Return Type: Investigator object.

Arguments: The following table displays the method's arguments.

Table 4–2 UpdateInvestigator Method Arguments

Argument Name	Type	Description
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
Investigator	String	A code for the Investigator; up to ten characters.
Title	String	The Investigator's title; for example, `Dr`.
First Name	String	The Investigator's given name.
Initials	String	The Investigator's initials.
Last Name	String	The Investigator's family name.
Phone Number	String	The Investigator's telephone number.
Address Name	String	The Investigator's institution.

Table 4–2 (Cont.) UpdateInvestigator Method Arguments

Argument Name	Type	Description
AddressLine1	String	The first line of the address of the Investigator.
AddressLine2	String	The second line of the address of the Investigator, if any.
AddressLine3	String	The third line of the address of the Investigator, if any.
City	String	The city in the Investigator's address.
State	String	The state must be a valid region code defined in the Regions form in Oracle Clinical.
Country	String	The country must be a valid region code defined in the Regions form in Oracle Clinical.
Postal Code	String	The postal code in the Investigator's address.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Request Investigator

This method returns the attributes of an Investigator in Oracle Clinical. If you pass the Oracle Clinical Investigator ID, Oracle Clinical uses that to identify the correct Investigator. If not, the system uses the Investigator code, which may have originated in the source system.

Web Service Name: InvestigatorService

Method Name: RequestInvestigator

Return Type: Investigator object.

Arguments: The following table displays the method's arguments.

Table 4–3 RequestInvestigator Method Arguments

Argument Name	Type	Description
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
Investigator	String	A code for the Investigator; up to ten characters.

Delete Investigator

This method deletes an Investigator from Oracle Clinical. If you pass the Oracle Clinical Investigator ID, Oracle Clinical uses that to identify the correct Investigator. If not, the system uses the Investigator code, which may have originated in the source system.

If the Investigator is assigned to a study site or if any patient data has been entered against the Investigator, the method does not delete the Investigator and returns an error.

Web Service Name: InvestigatorService

Method Name: DeleteInvestigator

Return Type: No return value. Raises an exception if there is an error.

Arguments: The following table displays the method's arguments.

Table 4–4 DeleteInvestigator Method Arguments

Argument Name	Type	Description
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
Investigator	String	A code for the Investigator; up to ten characters.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The hostname of the source database.

Site Service

The Site service allows you to create, update, request or delete a study site in Oracle Clinical.

SiteService has the following methods:

- "Create Site" on page 4-16
- "Update Site" on page 4-17
- "Request Site" on page 4-17
- "Delete Site" on page 4-18

Create Site

This method creates a new study site in Oracle Clinical. The study site is created as Active.

Web Service Name: SiteService

Method Name: CreateSite

Return Type: Site object with autopopulated values populated.

Arguments: The following table displays the method's arguments.

Table 4–5 CreateSite Method Arguments

Argument Name	Type	Description
Site	String	A unique code for the study site of ten characters or less. If the code is greater than ten characters, the system uses the generated Site ID instead.
Name	String	The name of the study site; up to 60 characters.
Phone Number	String	The study site's telephone number.
AddressLine1	String	The first line of the address.
AddressLine2	String	The second line of the address, if any.
AddressLine3	String	The third line of the address, if any.
City	String	The city in the address.
State	String	The state must be a valid region code defined in the Regions form in Oracle Clinical.
Country	String	The country must be a valid region code defined in the Regions form in Oracle Clinical.

Table 4–5 (Cont.) CreateSite Method Arguments

Argument Name	Type	Description
Postal Code	String	The postal code in the address.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The hostname of the source database.

Update Site

This method updates an existing study site in Oracle Clinical. The study site is set to Active. This method uses SiteId to identify the study site or, if that value is not passed in, it uses the value of study site. If no current study site matches, it raises an exception.

Web Service Name: SiteService

Method Name: CreateSite

Return Type: Site object.

Arguments: The following table displays the method's arguments.

Table 4–6 UpdateSite Method Arguments

Argument Name	Type	Description
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
Site	String	The unique site code; up to ten characters.
Name	String	The name of the study site; up to 60 characters.
Phone Number	String	The study site's telephone number.
AddressLine1	String	The first line of the address.
AddressLine2	String	The second line of the address, if any.
AddressLine3	String	The third line of the address, if any.
City	String	The city in the address.
State	String	The state must be a valid region code defined in the Regions form in Oracle Clinical.
Country	String	The country must be a valid region code defined in the Regions form in Oracle Clinical.
Postal Code	String	The postal code in the address.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The hostname of the source database.

Request Site

This method returns the attributes of a study site in Oracle Clinical. This method uses SiteId to identify the study site or, if that value is not passed in, it uses the value of study site.

Web Service Name: SiteService

Method Name: RequestSite

Return Type: Site view object.

Arguments: The following table displays the method's arguments.

Table 4–7 RequestSite Method Arguments

Argument Name	Type	Description
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
Site	String	The unique site code; up to ten characters.

Delete Site

This method deletes a study site from Oracle Clinical. If the study site is assigned to a study or if any patient data has been entered against it, the method does not delete the study site and returns an error.

Web Service Name: SiteService

Method Name: DeleteSite

Return Type: Void. Raises an exception if there is an error.

Arguments: The following table displays the method's arguments.

Table 4–8 DeleteSite Method Arguments

Argument Name	Type	Description
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
Site	String	The unique site code; up to ten characters.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Study Site Service

The Study Site service allows you to create, request, update or delete a study site. It also allows you to create, request, or delete the Investigator assignment for a study site.

InvestigatorService has the following methods:

- "Create Study Site" on page 4-19
- "Update Study Site" on page 4-19
- "Assign Investigator" on page 4-20
- "Request Current Investigator Assignment" on page 4-21
- "Request Investigator Assignment" on page 4-21
- "Delete Investigator Assignment" on page 4-22
- "Request Study Site" on page 4-22
- "Delete Study Site" on page 4-23

Create Study Site

This method creates a new study site in Oracle Clinical.

Web Service Name: StudySiteService

Method Name: CreateStudySite

Return Type: StudySite object with autopopulated values populated.

Arguments: The following table displays the method's arguments.

Table 4–9 CreateStudySite Method Arguments

Argument Name	Type	Description
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.
StudySite	String	The unique code to identify the site within the study; must be 10 characters or less.
StartDate	Date	The date the study site started participating in the study.
EndDate	Date	The date the study site stopped participating in the study.
FreezeFlag	Boolean	Frozen study sites cannot have data entered or changed. However, setting this flag through this Web service has no effect. To freeze data, use the batch job in Oracle Clinical under Conduct, Security, then Freeze.
OktoShipFlag	Boolean	If True, the site is approved for the shipment of drug supplies.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Update Study Site

This method updates an existing study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception.

Web Service Name: StudySiteService

Method Name: UpdateStudySite

Return Type: StudySite object.

Arguments: The following table displays the method's arguments.

Table 4–10 UpdateStudySite Method Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
ClinicalStudy	String	The unique study code; up to ten characters.

Table 4–10 (Cont.) UpdateStudySite Method Arguments

Argument Name	Type	Description
Site	String	The unique site code; up to ten characters.
StudySite	String	The unique code to identify the site within the study; must be 10 characters or less.
StartDate	Date	The date the study site started participating in the study.
EndDate	Date	The date the study site stopped participating in the study.
FreezeFlag	Boolean	Frozen study sites cannot have data entered or changed. However, setting this flag through this Web service has no effect. To freeze data, use the batch job in Oracle Clinical under Conduct, Security, then Freeze.
OktoShipFlag	Boolean	If True, the site is approved for the shipment of drug supplies.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Assign Investigator

This method assigns an existing Investigator to a study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception. If an Investigator is already assigned to the study site, the existing Investigator assignment is terminated with a termination date of sysdate.

Web Service Name: StudySiteService

Method Name: AssignInvestigator

Return Type: StudySiteRole object.

Arguments: The following table displays the method's arguments.

Table 4–11 AssignInvestigator Method Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the Site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
StartDate	Date	The date the Investigator was assigned to the study.
TerminationDate	Date	The date the Investigator was removed from the study.
ContractDate	Date	The date the Investigator's contract began.
DiscontinuationLetterDate	Date	The date the Investigator's letter of discontinuation was sent, if any.

Table 4–11 (Cont.) AssignInvestigator Method Arguments

Argument Name	Type	Description
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Request Current Investigator Assignment

This method returns the attributes of the current Investigator assignment to a study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception.

Web Service Name: StudySiteService

Method Name: RequestCurrentInvestigatorAssignment

Return Type: StudySiteRole object.

Arguments: The following table displays the method's arguments.

Table 4–12 RequestCurrentInvestigatorAssignment Method Arguments

Argument Name	Type	DescriptionDescription
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the Site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.

Request Investigator Assignment

This method returns the attributes of an existing Investigator assignment to a study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception.

Web Service Name: StudySiteService

Method Name: RequestInvestigatorAssignment

Return Type: StudySiteRole object.

Arguments: The following table displays the method's arguments.

Table 4–13 RequestInvestigatorAssignment Method Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.

Table 4–13 (Cont.) RequestInvestigatorAssignment Method Arguments

Argument Name	Type	Description
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
StartDate	Date	The date the Investigator was assigned to the study.

Delete Investigator Assignment

This method terminates the assignment of an Investigator to a study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception.

Web Service Name: StudySiteService

Method Name: DeleteInvestigatorAssignment

Return Type: No return value. Raises an exception if there is an error.

Arguments: The following table displays the method's arguments.

Table 4–14 DeleteInvestigatorAssignmentMethod Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the study site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.
InvestigatorId	Double	INVESTIGATOR_ID; the unique numeric database code for the Investigator.
TerminationDate	Date	The date the Investigator was removed from the study.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Request Study Site

This method returns the attributes of a study site in Oracle Clinical, including past and present Investigators assigned, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception.

Web Service Name: StudySiteService

Method Name: RequestStudySite

Return Type: StudySite with all StudySiteRoles.

Arguments: The following table displays the method's arguments.

Table 4–15 RequestStudySite Method Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the Site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.

Delete Study Site

This method deletes a study site in Oracle Clinical, using the ClinicalStudyId and SiteId to find the correct row. If those values are not passed in, it uses the values of ClinicalStudy and Site. If no matching row exists, the method raises an exception. If an Investigator is assigned to the study site or if any patient data has been entered against it, the method does not delete the study site and returns an error.

Web Service Name: StudySiteService

Method Name: DeleteStudySite

Return Type: No return value. Raises an exception if there is an error.

Arguments: The following table displays the method's arguments.

Table 4–16 DeleteStudySite Method Arguments

Argument Name	Type	Description
ClinicalStudyId	Double	CLINICAL_STUDY_ID; the unique numeric database code for the study.
SiteId	Double	SITE_ID; the unique numeric database code for the Site.
ClinicalStudy	String	The unique study code; up to ten characters.
Site	String	The unique site code; up to ten characters.
SourceApplication	String	The name of the application sending the message.
UserName	String	The user name of the person sending the message.
DatabaseInstance	String	The database instance ID of the system creating the investigator.

Partial Source Data Verification Web Service

You can use the Partial Source Data Verification (PSDV) web service to create a new patient SDV draft plan for a study site, update an existing draft plan, as well as publish the new plan immediately or defer publishing by saving it as a draft. Through this web service, you can update patient SDV plans in real time for a study site, rather than going through the RDC Onsite user interface.

> **Note:** Oracle recommends that you test the web service in Oracle Enterprise Manager (OEM) or by building a web service proxy before using it in a production environment. See Testing the PSDV Web Service for instructions to test this web service using OEM.

Web Service Name: DataPushService

Method Name: populateStudySiteDefaults

Return Type: PSDVResult object, with two fields:

- Result: `0` if successful, or `-1` if an error occurred
- Error: contains error details, if an error occurred, otherwise blank.

Arguments: The following table displays the method's arguments.

Table 4–17 PopulateStudySiteDefaults Method Arguments

Argument Name	Type	Description
Study	String	The name of the study for which the patient SDV plan is being created or updated.
StudySite	String	The name of the site within the study for which the patient SDV plan is being created or updated.
PatInitCount	Integer	The initial count of patients to be selected for SDV.
PatAutoSelRate	Integer	The percentage of patients to be automatically selected for SDV.
Datasource	String	Optional. The name of the datasource to be used by the web service. The name should be that of the default datasource defined in rdcConfig.properties. The file is in the OPA_CONFIG_FOLDER, whose location is specified in the opa51 registry under HKEY_LOCAL_MACHINE > Software > Oracle.
User	String	The application username that the web service uses to perform updates. The audit trail reflects changes made by the web service as being performed by the user provided here.
PublishAction	String	Indicates if the new or updated plan should be published or saved as a draft. The accepted values are: - `P` to publish the plan - `D` to save the plan as a draft. See Table 4–18 for details on how the value you set for this argument interacts with the planAction argument value.
PlanAction	String	Indicates which action is taken in relation to the plan, depending on its in-application status. The accepted values are: - `NA` to take no action if the plan already exists - `UD` to update the draft with the values passed, if the plan already exists - `RD` to recreate the already existing draft with the values passed. See Table 4–18 for details on how the value you set for this argument interacts with the publishAction argument value.

The combined use of the two action arguments, publishAction and planAction, can produce multiple outcomes, as shown in the following table.

Table 4–18 Outcomes of the Different Combinations of Publish and Plan Actions

#	Publish Action	Plan Action	Outcome If Draft Plan Exists	Outcome If Draft Plan Does Not Exist
1	D	NA	The existing draft plan is not changed.	A new draft plan is created with the values passed by the web service.
2	D	UD	The existing draft plan is updated with the new values passed by the web service.	A new draft plan is created with the values passed by the web service.
3	D	RD	A new draft plan is created as a copy of the published plan. The values for the initial patient count and auto-selection rate are passed by the web service. The entity level selection (patient or CRF) is inherited from the published plan. If no published plan exists, entity level selection is not defined for the new draft plan.	A new draft plan is created with the values passed by the web service.
4	P	NA	The web service makes no changes.*	A new plan is created with the values passed by the web service and the plan is published.
5	P	UD	The existing draft plan is updated with the new values passed by the web service and the plan is published.	A new plan is created with the values passed by the web service, and then published.
6	P	RD	A new draft plan is created as a copy of the published plan and is then published. The values for the initial patient count and auto-selection rate are passed by the web service. The entity level selection (patient or CRF) is inherited from the existing published plan. If no published plan exists, entity level selection is not defined for the new plan.	A new plan is created with the values passed by the web service, and then published.

* To publish an existing draft plan, use the RDC Onsite user interface. See *Creating a New SDV Plan* for instructions.

Testing the PSDV Web Service

You can test the PSDV method of the Data Push Service using the web services test UI in Oracle Enterprise Manager (OEM). Before you proceed, make sure that the OPA server is running on the server where the Java Key Store is configured and that RDC Onsite is deployed and running.

> **Note:** This section provides instructions to test web service and security implementation without generating a web service client, which is required to use the web service in a live environment.
>
> You can generate a web service client for deployment using a published WSDL. To locate the URL of the WSDL for the PSDV web service in the deployment, go to http://*appserver*:7221/rdcservice/DataPushService?wsdl, where *appserver* is replaced with the name of the server you are using.

1. Open OEM on the server where the Java Key Store is configured.

   The address for OEM is of the form http://*appserver*:7101/em, where *appserver* is replaced by the name of the application server you are using.

2. Log in to OEM using your credentials.
3. In the tree menu on the left, expand the **WebLogic Domain** option by clicking on the plus icon to its left. The OPADomain option is now visible beneath.
4. Right-click on **OPADomain** in the tree menu and go to **Web Services**, then select **Test Web Service**.
5. In the WSDL field, add the address of the web service WSDL file. The URL is of the form http://*appserver*:7221/rdcservice/DataPushService?wsdl, where *appserver* is replaced with the name of the server you are using.

   > **Note:** To identify the URL of the WSDL file for the Data Push Service, go to http://*appserver*:7221/rdcservice/DataPushService, where *appserver* is replaced with the name of the server you are using.

6. Click **Parse WSDL**. The details of the identified web service are retrieved and you should see the value populateStudySiteDefaults in the Operation drop-down list if you used the correct WSDL file.
7. In the Security section, select the **OWSM Security Policies** radio button.
8. From the Compatible Client Policies list, select **oracle/wss11_x509_token_with_message_protection_client_policy**.
9. Provide the location where the Java Key Store is configured in the JKS Keystore Location field. For example, C:\wss11_x509\client\client_keystore.jks.
10. In the JKS Keystore Password field, enter the password for the keystore above.
11. Click on **Load Keys** to the right of the fields to import signature key information from the keystore.
12. Select the **Advanced Options** box.
13. Select the **webservice** option from the Signature Key Alias drop-down list and then fill in the Signature Key Password field beneath with the password for the webservice alias.
14. In the Input Arguments section, set values for the arguments of the web service. All arguments are mandatory, except for Datasource.

    See Table 4–17 for information on each of the arguments.

15. Click on **Test Web Service** in the lower right corner of the screen to test the web service with the current settings.

Once the test runs, the Response tab opens. The result of the test is shown in the two fields, result and error. The result field contains the value 0 if the test was successful, or -1 if an error occurred. If an error occurred, the details are shown in the error field.

Part III

Interfacing with the Data Entry Subsystem

This section includes:

5

Data Capture API Overview

The term *API*, or *Application Programming Interface*, refers to the mechanisms that form a software interface to link an application program to the services and functions of another application, operating system, network operating system, driver, or software program.

The Data Capture API (DCAPI) provides API functions to update data in Oracle Clinical. In addition to the API, Oracle Clinical provides a stable interface for programmers. The combination of the Data Capture API and the stable interface provides the tools to integrate Oracle Clinical with other systems. For information about the stable interface contact Support or see the *Oracle Clinical Stable Interface Technical Reference Manual.*

Using the API

The API can be used with programs that you develop with Oracle Pro*C and Microsoft Visual C++. See the requirements section of the *Oracle Clinical Installation Guide* for the supported version Oracle Pro*C and Microsoft Visual C++. The following files are provided to use in your development, and are available on the Oracle Clinical Web Server in the *opa_home*/oc/dcapi folder. Please note that DCAPI binaries are compiled for 64-bit. This means that the bin folder contains all the 64-bit libraries and calling application need to be in 64-bit.

- `dcapi.h`: Include this header file in your C files and set the header file path in your Microsoft Visual C++ project definition to the directory that contains dcapi.h.
- `dcapi.lib`: Include this library file in the list of files used to link your program and set the library path of your Microsoft Visual C++ project definition to the directory that contains dcapi.lib.
- `dcapi.dll`: Include this file in your system path during runtime.

File Locations

The Oracle Clinical Installer automatically loads the DCAPI files onto the Oracle Clinical Web Server during the Front End installation. The DCAPI files are available in the *oracle_home*/oc/dcapi folder, and are required by Oracle Clinical. To link to these files in a different location, copy the DCAPI files to that location instead of moving them.

To build an application using DCAPI, the C++ compiler must know where the header and library files are, and what libraries to link against. The following section describes how to do this in Visual Studio 2010.

Setting Up Linkage in Microsoft Visual C++

To use the header file and library file in your Microsoft Visual C++ projects, within Visual Studio, right click on your project in the Solution Explorer and bring up the project's **Properties** dialog. You need to change the settings for both **Debug** and **Release** versions, so under the **Configuration** field, select "All Configurations". The following example shows how to do so in Microsoft Visual C++ 10.0; the steps may differ if you use another version.

Header File

Select the C/C++ category of properties, and pick the **General** options under that. In the **Additional Include Directories** property, add the semicolon-separated directory where dcapi.h is available

Static Library File

To change the settings for both Debug and Release versions, go to the Properties dialog under the Configuration field and select **All Configurations**.

Select the **Linker** category of properties then pick the **Input** option. In the **Additional Dependencies** property, add the DCAPI library `dcapi.lib`.

Sample C Programs

OPA provides a selection of sample programs that illustrate the use of DCAPI functions in the following tasks for a document:

- Initial Log-in
- Key Changes
- First-pass data entry
- Update
- Browse
- Soft Delete

This document is available on My Oracle Support. Search for article ID 315201.1 and click the dcapi_samples.pc link from that note; see "Finding Information on My Oracle Support" on page viii.

Auditing Considerations

Many DCAPI functions use two new data elements that provide auditing information: the audit change reason and a reason for the change. These data elements are part of the new AuditInfo structure, which is a direct parameter for the DCAPI functions SetResponseData, SetDataComment, and DeleteRepeat and an indirect parameter for the functions SetRdci and SetRdcm.

AuditInfo is also a new component of other existing data structures in the DCAPI, adding audit capabilities to these structures. All DCAPI functions that use these structures are affected by the change.

RdciKeysRecord: The external system will use AuditInfo to provide the audit change reason and an optional comment, in addition to the new set of keys or comment definition, in the RdciKeysRecord. The RdciKeysRecord structure will also include another boolean called audit_only_flag to indicate whether only an audit reason and no key or comment changes is provided. If an audit reason is required and it is not

provided and the system is unable to derive or default one, the API call will fail. This change affects the DCAPI functions CreateRdci and SetRdci.

RdciRecord: The external system uses AuditInfo to retrieve the audit reason and comment (if any) provided earlier during an RDCI key or comment change or a soft delete, in addition to the full set of RDCI elements, in the RdciRecord. Adding AuditInfo to this structure affects the functions CreateRdci, GetRdci, ProcessRdci, SetRdci and FetchRdci.

RdcmKeysRecord: This structure will be updated to include the AuditInfo structure. AuditInfo is the structure that the external system will use to provide audit reason and an optional comment, in addition to the new set of keys or comment definition, in the RdciKeysRecord. If an audit reason is required and it is not provided, and the system is unable to derive or default one, the API call will fail. This change affects the SetRdcm function.

RdcmRecord: This structure will be updated to include the AuditInfo structure. This is the structure that the external system will use to retrieve the audit reason and comment (if any) provided earlier during an RDCM key or comment change, in addition to the full set of RDCM elements, in the RdcmRecord. This change affects the functions GetRdcm and SetRdcm.

RepeatId: This structure will be updated to include the AuditInfo structure. The boolean needs_audit will be set to TRUE to indicate if a audit reason is needed and is missing. The external system uses this structure to provide the audit reason and an optional comment when deleting a repeat.

DataComment: This structure will be updated to include the AuditInfo structure. The external system uses AuditInfo to provide the audit reason and an optional comment, in addition to the investigator comment definition, in the DataComment. This change affects the following DCAPI functions: CreateRdci, SetRdci, GetRdci, FetchRdci, ProcessRdci, SetRdcm, GetRdcm, DeleteRepeat and SetDataComment.

The values to these new parameters will be initialized to NULL and will be cleared upon saving of a transaction.

Auditing and Backward Compatibility

DCAPI functions require an audit change reason each time you make a change to a document. To enable backward compatibility with programs you have written to interface with earlier Oracle Clinical releases, this release includes two new reference codelists that you can supply with default values for the audit change reason.

To enable backward compatibility for pre-4.5 interfacing programs, perform these steps:

1. Re-compile and re-link your existing interfacing programs with the Release 4.5 version of the header file (dcapi.h) and library file (dcapi.lib).
2. Populate the new reference codelists with the default values you want to use for the audit change reason. You can Because the audit change reason supplied depends on the change you make, you can provide separate default values for:
   - Key or Comments Changes to RDCIs/RDCMs, and Soft-deletions of Documents
   - Investigator Comment Changes
   - Deletion of Repeats

Key or Comments Changes to RDCIs/RDCMs, and Soft-deletions of Documents

To find the default audit reason for Header changes (RDCI or RDCM Key change, RDCI or RDCM Comment change and when Soft-Deleting a document), the system refers to the reference code list 'RDCI CHANGE REASON TYPE CODE' as the primary and to 'RDCI CHANGE REASON2 TYPE CODE' as the secondary reference codelist. The system attempts to find a default audit reason by looking for "DCAPI" as a comma-separated string in the long value of either one of the reference codelists.

Investigator Comment Changes

To find a default audit reason for Investigator Comment changes (a change to responses), the system refers to the reference code list 'DATA CHANGE REASON TYPE CODE' as the primary and to 'DATA CHANGE REASON2 TYPE CODE' as the secondary reference codelist. The system attempts to find a default audit reason by looking for "DCAPIINV" in the long value of either one of the reference codelists.

Deletion of Repeats

To find a default audit reason for Delete Repeat (a change to responses), the system refers to the reference code list 'DATA CHANGE REASON TYPE CODE' as the primary and to 'DATA CHANGE REASON2 TYPE CODE' as the secondary reference codelist. The system attempts to find a default audit reason by looking for "DCAPIDEL" in the long value of the either one of the reference codelists.

About the Data Capture API Documentation

This documentation provides information about the syntax and functionality of the API functions for creating or updating the following information in Oracle Clinical:

- received DCI and DCM Log-In information
- actual events
- patient enrollment
- data responses
- data discrepancies
- comment information

This chapter is an introduction to Chapter 7, which describes the functions themselves.

Special Notes About the API

Review the following rules before you begin to develop your application. Instructions are written for someone with a working knowledge of Microsoft Visual C++ 6.0.

Accessible Documents

The documentation includes several references to the term *accessible documents*. Every Oracle Clinical study has properties indicating if first and second pass Data Entry are necessary. A RDCI is considered accessible when any of its RDCMs become accessible.

An RDCM is considered accessible if it meets any of the following criteria:

- The RDCM was batch loaded
- Pass 1 has been completed for the RDCM and Pass 2 is not required
- Pass 2 is required for the RDCM and has been completed

Date Formats

All date, time and date-time arguments to the API function calls are of type character array. This information is translated to Oracle dates internally by the API functions, so to remove ambiguity during translation, the dates, times, and date-times must be input in the following formats: YYYYMMDD, HH24MISS, and YYYYMMDDHH24MISS. The hours, HH, are specified in military time (values from 00 to 23 inclusively). These formats are referred to as the API date, time, and date-time formats respectively.

Error Handling

API functions will return a SUCCESS, FAILURE, or WARNING, defined as `"short int"` in the header file (SUCCESS = 0, FAILURE = 1, WARNING = 2). In the case of FAILURE or WARNING, the external application can use the GetError API function to get the message ID, text, and type from the error message stack. The external application can also use GetErrorStackSize to assess the depth of the stack and then call GetError repeatedly until no errors remain. If GetError returns SUCCESS your application should process the errors. For a complete list of error messages, see Appendix A, "Error Messages". If GetError returns the message ID -1, an internal Oracle Clinical error has occurred.

Warning: An internal Oracle Clinical error can corrupt data. If you encounter this kind of error, immediately roll back your data. The external application should continue retrieving the errors and ignoring them until the error stack is empty.

Indicating NULL for Number Fields

Your application should put -1 in numeric (double, short, int) fields to indicate a null value to Oracle Clinical. Similarly, Oracle Clinical returns -1 for these fields to indicate a null value in the database. The Visit Number field can have a valid negative value. If the visit number has a value of -1, your application must check for a visit name. When the visit name is null and the visit number is -1, the visit number is null in the database as well.

Overall Memory Allocation and Management

The API functions described manage allocation and deallocation of memory for the internal data structures they use. Your application must manage allocation and deallocation of memory for arguments to API calls that accept pointers to variables. For example, in a call to GetRdci, your program must allocate the memory for the argument of type `RdciRecord`.

Memory Allocation Rules for Data Entry

InitializeRdcmResponses handles memory allocation. If you attempt to set or get response values for a question in a particular repeat in a question group without creating a repeat first, you receive an error message.

For a document with existing responses, InitializeRdcmResponses allocates memory to hold response values for all questions in all existing repeats in all question groups in the RDCM. The user can then retrieve or set values for these responses. However, for repeating groups, if the user wants to retrieve or set the response of a new repeat that does not exist in the database, the user must begin by inserting a repeat.

For repeating groups with `n > 1` repeats, InsertRepeat needs to be called. This function allocates memory to hold response values for all questions in the *i*-th repeat for the group (where `1 < i <= n`).

It is not necessary to enter response values immediately after inserting the repeat; the user of the API functions can first insert the necessary repeats in all question groups and then set the values for the responses. However, response values for at least one question should be entered for all inserted repeats in all question groups, otherwise WriteResponses will return FAILURE.

To delete an existing repeat, use the DeleteRepeat call. This does not free the memory allocated for holding the responses in the API Responses buffer; it simply marks the corresponding responses as deleted. They will be processed at commit time and marked accordingly in the database. The deleted repeats are archived in the database. Users can run either an Oracle Clinical audit report or their own audit report to view the deleted repeats.

Procedure Classification

Before you can use the ExecuteMultivariate function, Validation and Derivation Procedures must be categorized in Oracle Clinical. This step lets the function, ExecuteMultivariate identify the Procedures to execute in different circumstances.

Procedures are classified by the Execution Context attribute. A field for the attribute is displayed in the Oracle Clinical Procedures forms. Although this attribute can be entered for both pre-V3.1- and V3.1-style Procedures, only V3.1-style Procedures can be called by ExecuteMultivariate. The Execution Context does not affect batch validation processing.

This attribute's value should be chosen based on the Procedure's purpose and its usefulness in raising multivariate discrepancies for timely feedback during data entry. While the value defaults to OFF-LINE, V3.1-style Procedures can change it to either ON-LINE/DCM or ON-LINE.

Using Execution Context

These sections describe the values for the Execution Context attribute. An API-only argument, SINGLE, is also described because it affects Procedure classification.

ON-LINE/DCM

Procedures should be classified as ON-LINE/DCM when they are suitable for DCM-targeted types of validation to be invoked— for instance, at the completion of data entry for a given patient's DCM. Procedures with aggregate-only question groups, which have no primary reference DCM, cannot be classified as ON-LINE/DCM.

Calling ExecuteMultivariate with an ON-LINE/DCM Execution Context causes a specific DCM to be supplied in the API call. Procedures are then targeted for execution on the basis of a match on:

- Execution Context
- The Procedure primary reference question group DCM

Note: When ExecuteMultivariate is called with an ON-LINE Execution Context, Procedures classified as ON-LINE/DCM are automatically executed, regardless of their primary reference question group DCM.

ON-LINE

An ON-LINE classification indicates the Procedure is suited to being invoked after a larger unit of work (two or more DCMs) is completed for the patient — regardless of the Procedure primary reference question group DCM. Procedures classified as ON-LINE or ON-LINE/DCM are automatically executed when ExecuteMultivariate is called with an ON-LINE Execution Context.

OFF-LINE

Procedures classified as OFF-LINE (the default) are not executed when the API Execution Context argument is either ON-LINE or ON-LINE/DCM. They are typically appropriate for the following circumstances:

- batch processing
- execution in batch validation
- execution via the **Execute Single Procedure** menu option

SINGLE

An API-only argument, SINGLE, signifies that a specific Procedure is to be executed; the Procedure ID and version are supplied as API arguments in this case. When ExecuteMultivariate is called with the Execution Context of SINGLE, the match on Procedure classification is bypassed and the requested Procedure executed.

Relationship Between Procedure Definition and Procedure Execution

Procedures can have dependencies, such as using the results of derivation for subsequent validation. The Procedure Definition form enforces the restriction that a Procedure can only be flagged for a particular Execution Context if all Procedures upon which it is dependent are also in the same context or in a "lesser" context. This means that "related" Procedures must be guaranteed to execute in tandem.

The following tables shows, for six sample Procedures, whether the Procedure will be selected for execution in each context.

Procedure Definition	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — Off-Line Processing	Procedure Execution — Off-Line Processing
	ON-LINE/ DCM DEMOG	ON-LINE	Single Procedure B	Batch Validation	Execute Single Procedure B
Proc: A, **Primary Ref DCM**: DEMOG, **Execution Context**: ON-LINE/DCM	Selected	Selected	Not selected	Selected	Not selected

Procedure Definition	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — On-Line Processing API Execution Context	Procedure Execution — Off-Line Processing	Procedure Execution — Off-Line Processing
Proc: B, **Primary Ref DCM**: DCM48, **Execution Context**: ON-LINE/	Not selected	Selected	Selected	Selected	Selected
Proc: C, **Primary Ref DCM**: DCM48, **Execution Context**: OFF-LINE/	Not selected	Not selected	Not selected	Selected	Not selected
Proc: D, **Primary Ref DCM**: DEMOG, **Execution Context**: ON-LINE/DCM	Selected	Selected	Not selected	Selected	Not selected
Proc: E, **Primary Ref DCM**: VITAL_SIGNS, **Execution Context**: OFF-LINE	Not selected	Not selected	Not selected	Selected	Not selected
Proc: F, **Primary Ref DCM**: VITAL_SIGNS, **Execution Context**: ON-LINE	Not selected	Selected	Not selected	Selected	Not selected

For instance, if Procedure X uses derived values from Procedure Y the possible classification combinations between the two Procedures are shown below.

	Validation Procedure X Definition		
Derivation Procedure Y Definition	**ON-LINE/ DCM**	**ON-LINE**	**OFF-LINE**
on-line/DCM	Selected	Selected	Selected
on-line	Not selected	Selected	Selected
off-line	Not selected	Not selected	Selected

Security Access for the API

Oracle Clinical provides a mechanism for controlling the tasks that certain users can perform. Each user with access to Oracle Clinical is given a set of roles that ensures that only certain tasks can be performed by that user. A user's assigned roles are the same for each clinical study to which he or she has access.

The Data Capture API provides similar security mechanisms for assigning roles to users to determine the tasks they can perform. This assignment is also accomplished

through database roles. The roles necessary to perform tasks through the API are different from the roles in Oracle Clinical. Therefore, a particular user may be granted permission to perform data entry through an application built with the API but not through the Oracle Clinical client, or vice versa.

Database Roles for the API

The following list shows the names of the database roles and what each role is called in Oracle Clinical user materials.

Database Role Name	OC Manual Name
OCLAPI_INITIAL_LOGIN	Initial Log-In
OCLAPI_KEY_CHANGES	Key Changes
OCLAPI_FIRST_PASS_ENTRY	First-Pass Entry
OCLAPI_UPDATE	Update
OCLAPI_BROWSE	Browse
OCLAPI_PATIENT_ENROLLMENT	Patient Enrollment
OCLAPI_EXECUTE_MULTIVARIATE	Execute Multivariate

This section shows which of the database roles for the Data Capture API a user must have to act on production data. The column headers indicate different DCAPI roles. The rows indicate different DCAPI calls under different conditions. An "X" in a column signifies that the call in the selected row requires one of the roles indicated. Privileges are cumulative and require that the user have only one intersecting role/call combination for the call to succeed.

> **Note:** So that these actions are performed on test data, the letter T is appended to the database role name — for example, OCLAPI_BROWSE becomes OCLAPI_BROWSET.

Each role maps to a specific action that you can take within Oracle Clinical. This action is referred to as a *mode*. There are two types of modes:

Standalone - Patient Enrollment and Multivariate Execution are standalone roles.

Data Entry - Initial Log-In, Key Changes, First-Pass Entry, Update, and Browse are data entry modes.

CreateRdci (INITIAL_LOGIN)

Requires: Initial Log-In only

CreateRdci (KEY_CHANGES)

Requires: Key Changes

FetchRdci (Locking) (INITIAL_LOGIN)

Requires: Initial Log-In for documents that are inaccessible, or accessible documents whose only updateable field is the comment field

FetchRdci (Locking) (KEY_CHANGES)

Requires: Key Changes

FetchRdci (Locking) (FIRST_PASS_ENTRY)
Requires: First-Pass Entry for inaccessible documents only.

FetchRdci (Locking) (UPDATE)
Requires: Update

FetchRdci (Locking) (BROWSE)
Requires: Browse, but only if the operator comment in Browse mode is enabled

FetchRdci (Non-Locking)
Requires: Initial Log-In, Key Changes, First-Pass Entry, Update, and Browse.

SetRdci
Requires: Initial Log-In (Update of the comment fields only), Key Changes, First-Pass Entry (Update of the comment fields and Blank Flag only), and Update (Update of the comment fields only).

SetRdcm
Requires: Initial Log-In (Update of the comment fields only), Key Changes, First-Pass Entry (Update of the comment fields and Blank Flag only), and Update (Update of the comment fields only).

WriteRdciRdcm
Requires: Initial Log-In (Update of the comment fields only), Key Changes, First-Pass Entry (Update of the comment fields and Blank Flag only), and Update (Update of the comment fields only).

DeleteRdci
Requires: Initial Log-In (Inaccessible documents only)

InitializeRdcmResponses (FIRST_PASS_ENTRY)
Requires: First-Pass Entry

InitializeRdcmResponses (UPDATE)
Requires: Update

InitializeRdcmResponses (BROWSE)
Requires: Browse

SetActualEvent
Requires: Initial Log-In and Key Changes

SetPageStatus
Requires: Initial Log-In, Key Changes, First-Pass Entry, and Update

EnrollPatient
Requires: Patient Enrollment

ExecuteMultivariate
Requires: Multivariate Validation

The FetchRdci function retrieves data from the Oracle Clinical database. To call FetchRdci, program must know the RDCI_ID in the Oracle Clinical database for the RDCI it wants to fetch. You can obtain this information through the stable interface.

The Data Capture API buffers the RDCI, RDCM, and response data so that your program can pass data to the API and validate it without committing the data to the database. When program is ready to commit the RDCI and RDCM data to the database, it calls WriteRdciRdcm. To commit response data to the database call WriteResponses.

CreateRdci, ProcessRdci, SetRdci, SetRdcm and SetResponseData perform one or more of the following:

- validation checks
- defaulting of fields
- setting of timestamps

Your program should refresh any data structures it created to store RDCI and RDCM information with the modified data structures passed back from these API functions. This keeps the data structures in your program consistent with the API buffered data.

Calling API Functions

The Data Capture API enforces rules about when you can call an API function. These rules are associated with the state of the data. The API considers the data to be in one of the following states.

Pre-Connection

This state occurs when the your application is running but not connected to the Oracle Clinical database. The API function ConnectOCL connects you to the Oracle Clinical database. All other Data Capture API functions require a connection to the Oracle Clinical database.

If you call a Data Capture API function, and you are not connected to the database, you receive an error message. Once connected to the Oracle Clinical database, you can disconnect or set study context from any of the remaining four states.

Pre-Study

This state occurs immediately after your program connects to the Oracle Clinical database and means that the study for which you are planning to enter data is unknown to the API. The SetStudyContext API function provides this context information. Any of the API functions that attempt to retrieve or set data require the study context and return an error if called in this state. Because SetExternalContext does not affect Oracle Clinical data, it can be called in this state.

In-Study with Empty RDCI/RDCM Buffer

This state occurs when the program has set the study context but has not called any API functions that update RDCI/RDCM data. RDCI/RDCM data must always be entered before response data because responses are associated with RDCMs that belong to RDCIs.

To proceed with entering RDCI/RDCM and responses data, you must either create a new RDCI by calling CreateRdci or fetch an existing RDCI to browse or update by calling FetchRdci. Either of these two functions will switch you into the RDCIRDCMWork state. There are no pending changes to RDCI/RDCM data or responses data in this state, so APIs that commit data to the database — EnrollPatient,

SetActualEvent, SetPageStatus, and ExecuteMultivariate — can be called. You can call SetExternalContext from this state as well.

RDCIRDCMWork

This state occurs when the program is retrieving or updating RDCI/RDCM data; while in this state, you cannot call functions that retrieve or update RDCI/RDCM data. The function SetExternalContext does not commit data or retrieve/update responses data, so it can be called in this state.

Pending changes must be saved to the database, using WriteRdciRdcm; otherwise they must be discarded using FlushRdciRdcm before more response data can be entered. InitializeRdcmResponses then brings you to the ResponsesWork state.

Once all pending changes have been saved or discarded, the program can call EnrollPatient, SetActualEvent, SetPageStatus, and ExecuteMultivariate to commit data. These API functions do not commit data if there are pending changes, because incomplete RDCI/RDCM information would cause data corruption.

ResponsesWork

This state occurs when your program is retrieving or updating response data. The API functions that update the RDCI/RDCM data cannot be called in this state.

Once all pending changes have been saved (using WriteResponses) or discarded (using FlushResponses), the program can commit data by calling EnrollPatient, SetActualEvent, SetPageStatus, and ExecuteMultivariate. These functions cannot be called if there are pending changes, because the functions would commit incomplete responses information to the Oracle Clinical database which would cause data corruption. The program can return to either the RDCIRDCMWork state or In-Study with Empty RDCIRDCM Buffer state by calling, saving, or discarding any pending changes.

Functional State Mapping Information

This section describes the functional states in which you can call each of the DCAPI functions. Because ConnectOCL is associated with only one state, the Pre-Connection state, this function is not listed.

CreateRdci

Can be called in:

- In-Study with Empty RDCI/RDCM Buffer.
- RDCI/RDCM Work, but only if there are no changes pending. If an audit comment is required and has not been provided before you call CreateRdci in this state, it returns an error.

DeleteRdci

Can be called in RDCI/RDCM Work, but only if there are no changes pending. If an audit comment is required and has not been provided before you call DeleteRdci in this state, it returns an error.

DeleteRepeat

Can be called in Responses Work. If an audit comment is required and has not been provided before you call DeleteRepeat in this state, it returns an error.

DisconnectOCL

Can be called in:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending
- Responses Work, but only if there are no changes pending

EnrollPatient

Can be called in:

- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending. Even if there are no changes pending, EnrollPatient cannot be called in this state if the RDCI was fetched for update until the database lock is released.
- Responses Work, but only if there are no changes pending. Even if there are no changes pending, EnrollPatient cannot be called in this state if the RDCI was fetched for update until the database lock is released.

ExecuteMultivariate

Can be called in:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending
- Responses Work, but only if there are no changes pending

FetchRdci

Can be called in:

- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending. If an audit comment is required and has not been provided before you call FetchRdci, the system returns an error.

FlushRdciRdcm

Can be called in RDCI/RDCM Work

FlushResponses

Can be called in Responses Work. If an audit comment is required and has not been provided before you call FlushResponses, the system returns an error.

GetManualDiscrepancy

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetManualDiscrepancy, the system returns an error.

GetNumRows

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetNumRows, the system returns an error.

GetQuestGroupId

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetQuestGroupId, the system returns an error.

GetQuestionId

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetQuestionId, the system returns an error.

GetRdci

Can be called in RDCI/RDCM Work.

GetRdcm

Can be called in RDCI/RDCM Work, but only if there are no pending RDCI changes. If pending RDCI changes exist, process them by calling ProcessRdci.

GetRdcmArr

Can be called in RDCI/RDCM Work, but only if there are no pending RDCI changes. If pending RDCI changes exist, process them by calling ProcessRdci.

GetResponse

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetResponse, the system returns an error.

GetUnivDiscrepancy

Can be called in Responses Work. If an audit comment is required and has not been provided before you call GetUnivDiscrepancy, the system returns an error.

InitializeRdcmResponses

Can be called in RDCI/RDCM Work

InsertRepeat

Can be called in Responses Work. If an audit comment is required and has not been provided before you call InsertRepeat, the system returns an error.

ProcessRdci

Can be called in RDCI/RDCM Work

ProcessRdcm

Can be called in RDCI/RDCM Work, but only if there are no pending RDCI changes. If pending RDCI changes exist, process them by calling ProcessRdci.

SetActualEvent

Can be called in:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending. Even if there are no changes pending, this API function cannot be called in this state if the RDCI was fetched for update until the database lock is released.

- Responses Work, but only if there are no changes pending. Even if there are no changes pending, this API function cannot be called in this state if the RDCI was fetched for update until the database lock is released.

SetDataComment

Can be called in Responses Work. If an audit comment is required and has not been provided before you call this API function, the function will return an error.

SetExternalContext

Can be called in:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending.

SetManualDiscrepancy

Can be called from Responses Work. If an audit comment is required and has not been provided before you call this API function, the function will return an error.

SetMiscDiscrepancy

Can be called from Responses Work. If an audit comment is required and has not been provided before you call this API function, the function will return an error.

SetPageStatus

Can be called from:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer
- RDCI/RDCM Work, but only if there are no changes pending. Even if there are no changes pending, this API function cannot be called in this state if the RDCI was fetched for update until the database lock is released.
- Responses Work, but only if there are no changes pending. Even if there are no changes pending, this API function cannot be called in this state if the RDCI was fetched for update until the database lock is released.

SetRdci

Can be called from RDCI/RDCM Work

SetRdcm

Can be called from RDCI/RDCM Work, but only if there are no pending RDCI changes. You can process these pending changes by calling ProcessRdci.

SetResponseData

Can be called from Responses Work

SetStudyContext

Can be called from:

- Pre-Study
- In-Study with Empty RDCI/RDCM Buffer

- RDCI/RDCM Work, but only if there are no changes pending.
- Responses Work, but only if there are no changes pending.

SetUnivDiscrepancy

Can be called from Responses Work. If an audit comment is required and has not been provided before you call SetUnivDiscrepancy, it returns an error.

WriteRdciRdcm

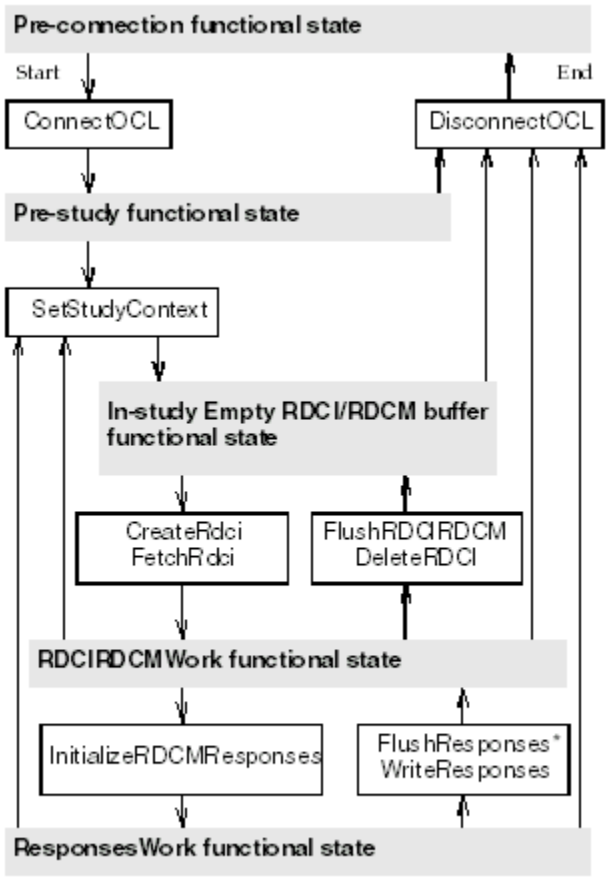
Can be called from RDCI/RDCM Work, but only if there are no pending RDCI changes. You can process these pending changes by calling ProcessRdci.

WriteResponses

Can be called from Responses Work. If an audit comment is required and has not been provided before you call this API function, the function will return an error.

Functional State Transition Diagram

The following diagram illustrates the required transitions to move from one state to another in the API.



This graphic shows which functions enable you to make a translation between functional states. From the Pre-connection state, run ConnectOCL to reach the Pre-study functional state. From Pre-Study, you can either run SetStudyContext to

reach the In-study Empty RDCI/RDCM buffer functional state. From In-study Empty RDCI/RDCM buffer, you can run CreateRdci or FetchRdci to reach the RDCIRDCMWork functional state. From RDCIRDCMWork, you can run FlushRDCIRDCM or DeleteRDCI to return to In-study Empty RDCI/RDCM buffer, or run InitializeRDCMResponses to reach ResponsesWork functional state. From ResponsesWork, you can run FlushResponses or WriteResponses to return to RDCIRDCMWork state. From any functional state except pre-connection, you can issue DisconnectOCL to disconnect.

****

* If called with preserve lock, otherwise brings you to In-Study with Empty RDCIRDCM Buffer state.

Data Management Rules

The majority of the Oracle Clinical data management rules are built into the functionality of the API. Nonetheless, there are a few data management rules that you must be aware of as you design your front-end application; otherwise, you may generate one or more error messages.

Five data entry modes are valid within the API. The following table describes the appropriate document status for each Log-In and Data Entry mode that the API supports.

Mode	Status
Initial Log-In	RECEIVED
Key Changes	RECEIVED, P1 STARTED, P1 COMPLETE
Update/Browse	P1 STARTED, P1 COMPLETE
P1 (First Pass)	RECEIVED, P1 STARTED, P1 COMPLETE only if document is not accessible and the operator is the person who created the document

Note that the Data Capture API does not support Pass 2. However, in some cases the status does change:

- If you update an RDCM in status PASS 2 STARTED or PASS 2 COMPLETE, the document is opened in UPDATE mode and after a save, the status of the document (RDCM and RDCI) remains unchanged.
- However, if you update an RDCM in status PASS 2 PENDING, then after a save, the RDCM status changes to PASS 2 COMPLETE. If all RDCMs in the RDCI are PASS 2 COMPLETE, then the RDCI status is also updated to PASS 2 COMPLETE.

6

DCAPI Structure Type Definitions

This section describes each of the data structures that are used by many of the DCAPI functions. For each structure, the section includes the syntax of the typedef declaration in dcapi.h, the functions and other structures that use this structure, and additional comments about some parameters.

ActualEventsRecord

Syntax

```
typedef struct actual_events_record
{
    char comment[COMMENT_SIZE + 1] ;
 } ActualEventsRecord;
```

Parameters

comment (in)

Used by DCAPI Function

SetActualEvent

AuditInfo

This structure is used to audit changes made to a document for RDCIs, RDCMs, or response changes. Three functions use AuditInfo as a direct parameter, and two others use it indirectly by using structures that contain AuditInfo as a parameter. These functions are:

Direct: SetResponseData, SetDataComment, DeleteRepeat

Indirect: SetRdci and SetRdcm

For instructions on using the AuditInfo structure in conjunction with DeleteRdci, see the section for that function on page 7-8.

Syntax

```
typedef struct audit_info
{
   char audit_comment[COMMENT_SIZE+1];
   char reason[AUDIT_REASON_SIZE+1];
   DCAPIFlag  needs_audit ;
} AuditInfo;
```

Parameters

audit_comment (in/out): This parameter provides a free-text description of the transaction. It could be defaulted by the system or can be user-provided.

reason (in/out): The audit reason parameter must match a value in the reference codelists RDCI CHANGE REASON TYPE CODE or RDCI CHANGE REASON2 TYPE CODE (for RDCI and/or RDCM change or for soft deletion of a document) and DATA CHANGE REASON TYPE CODE or DATA CHANGE REASON2 TYPE CODE (for response changes). This parameter could be defaulted by the system or can be user-provided.

needs_audit (out): This parameter indicates after the function call whether the audit reason was expected but was not provided. If TRUE, an audit reason was required but was not manually provided and the system could not supply a default reason.

Used by Functions

SetResponseData, SetDataComment, DeleteRepeat, SetRdci, and SetRdcm

DataComment

Syntax

```
typedef struct data_comm
{
   char data_comment[COMMENT_SIZE + 1];
   AuditInfo   audit ;
}DataComment;
```

Parameters

data_comment (in)

audit (in): A parameter of type AuditInfo, which tracks audit information about the investigator comment.

Used in Functions

SetDataComment

DiscInfo

If used by SetResponseData, GetUnivDiscrepancy, or GetManualDiscrepancy, the parameters will be populated by the system. If used by SetUnivDiscrepancy or SetManualDiscrepancy, the parameters will be entered manually.

Syntax

```
typedef struct disc_info
{
    char disc_type[DISC_TYPE_SIZE+1];
    char disc_rev_status[DISC_STATUS_SIZE+1];
    char disc_resolu_type[CODE_SIZE+1];
    char comment_text[EDMS_COMMENT_SIZE+1];
    char internal_comment_text[EDMS_COMMENT_SIZE+1] ;
    char resolution[EDMS_COMMENT_SIZE+1] ;
} DiscInfo;
```

Parameters

The following are input parameters when DiscInfo is used by the functions SetUnivDiscrepancy and SetManualDiscrepancy; they are output parameters when used by GetUnivDiscrepancy, GetManualDiscrepancy, SetMiscDiscrepancy, and SetResponseData.

disc_type (in/out): This parameter maps to the reference codelist MANUAL SOURCE TYPE CODE for manual discrepancies. For univariate discrepancies, this parameter is system-generated and should not be changed.

disc_rev_status (in/out): This parameter maps to the installation reference codelist DISCREPANCY REV STATUS CODE.

disc_resolu_type (in/out): This parameter maps to the reference codelist DISCREPANCY RESOLU TYPE CODE.

comment_text (in/out): For univariate discrepancies, the initial value may be system-generated if a corresponding EDMS setup has been already been done. However, this value can be updated. For manual discrepancies, this parameter is a free-text description.

internal_comment_text (in/out): Free text description.

resolution (in/out): Free text description.

Used by Functions

GetUnivDiscrepancy, GetManualDiscrepancy, SetResponseData, SetUnivDiscrepancy, SetManualDiscrepancy, and SetMiscDiscrepancy

ErrorInfo

The ErrorInfo structure is a standalone element that DCAPI functions use to retrieve errors from the stack.

Syntax

```
typedef struct error_info
{
    short int type;
    int message_id;
    char message_string[COMMENT_SIZE+1];
    char function_name[51];
} ErrorInfo;
```

Parameters

type (out): The severity; either ERR (error) or WRN (warning).

message_id (out): the error message ID.

message_string (out): the error message.

function_name (out): the API function.

Used by Functions

All DCAPI functions

ExternalContextInfo

This structure is used by SetExternalContext to track changes made to data.

Syntax

```
typedef struct context_info
{
   char userid[31];
   char timestamp[DATE_TIME_SIZE + 1];
   char trans_type[61];
   char comment[COMMENT_SIZE + 1];
}ExternalContextInfo;
```

Parameters

userid (in): the Oracle user who is entering or modifying data.

timestamp (in): date and time of transaction.

trans_type (in): the type of transaction, which must match a value in the EXTERNAL_TRANS_TYPE reference codelist.

comment (in): free-text description of the transaction.

Used by Function

SetExternalContext

GroupId

This structure is used by GetQuestGroupId to return the question group ID, given the question group name.

Syntax

```
typedef struct group_id
{
 char group_name[GROUP_SIZE+1] ;
} GroupId;
```

Parameter

group_name (in): the name of the question group.

Used in Function

GetQuestGroupId

MvInfo

This structure is used by ExecuteMultivariate to specify the criteria for multivariate validation submission.

Syntax

```
typedef struct mv_info
{
     char execution_context[16];
     double procedure_id ;
     int procedure_version_sn ;
     double patient_position_id ;
     double dcm_id ;
     double clinical_study_id ;
     double clinical_study_version_id ;
     double received_dci_id;
     short int mode ;
     DCAPIFlag debug_flag ;
} MvInfo;
```

Parameters

execution_context (in): Values are ON-LINE/DCM, ON-LINE or SINGLE

procedure_id (in)

procedure_version_sn (in)

patient_position_id (in)

dcm_id (in)

clincial_study_id (in)

clincial_study_version_id (in)

received_dci_id (in)

mode (in): Values are OCL_TEST or OCL_PROD.

debug_flag (in): Values are TRUE or FALSE.

Used in Function

ExecuteMultivariate

PatientRecord

This structure is used by EnrollPatient to enroll a patient in a study.

Syntax

```
typedef struct patient_record
{
    char birth_date[DATE_SIZE +1];
    char sex[2];
    char first_name[16];
    char last_name[21];
    char initials[5];
    char enrollment_date[DATE_SIZE+1];
} PatientRecord;
```

Parameters

birth_date (in)

sex (in)

first_name (in)

last_name (in)

initials (in)

enrollment_date (in)

Used in Function

EnrollPatient

QuestionId

The QuestionId structure is used by GetQuestionId to return the question ID, given the question group name, question name, and occurrence subset number.

Syntax

```
typedef struct question_id
{
    char group_name[GROUP_SIZE+1];
    char question_name[QUESTION_SIZE+1] ;
    int qn_occurence_sn;
} QuestionId;
```

Parameters

group_name (in): the name of its question group.

question_name (in): the name of the question.

qn_occurrence_sn (in): the occurrence subset number.

Used in Function

GetQuestionId

RdciKeysRecord

This structure contains the key values of the RDCI record being created or modified by CreateRdci or SetRdci.

Syntax

```
typedef struct rdci_keys_record
{
    char document_number[21] ;
    char patient[11] ;
    char dci_short_name[11] ;
    char dci_date[DATE_SIZE + 1] ;
    char dci_time[TIME_SIZE + 1] ;
    char visit_name[21] ;
    short int  subevent_number ;
    char site[11] ;
    char investigator[11] ;
    DCAPIFlag blank_flag ;
    char comment[COMMENT_SIZE + 1] ;
    AuditInfo   audit ;
    DCAPIFlag audit_only_flag;
} RdciKeysRecord;
```

Parameters

document_number (in): Optional for CreateRdci, required for SetRdci.

patient (in): Any unique identifier which the external system can assign to a patient. It is not necessarily the same as the 'patient position id' or the 'patient name'.

dci_short_name (in)

dci_date (in): Can be optional if the DCI does not require a date as part of its definition.

dci_time (in): Only applicable if the current DCI needs them to be collected; otherwise this parameter must be passed as null values. Passing a non-null value for this parameter when it is not applicable will cause the API function to fail. A null value for a character array field is the null ("") string.

visit_name (in)

subevent_number (in)

site (in): Any unique identifier which the external system can assign to a site.It is not necessarily same as the 'site id' or the 'site name'.

investigator (in): Any unique identifier that the external system can assign to an investigator. It is not necessarily the same as the investigator ID or the investigator name.

blank_flag (in): Value passed in are TRUE or FALSE.

comment (in): Free-text description.

audit (in): A parameter of type AuditInfo, which tracks audit information about the RDCI key or comment changes.

audit_only_flag (in): This parameter, if set to TRUE, indicates that the audit reason provided is for soft-deleting a document (via DeleteRdci), rather than for a key or comment change.

Used in Functions

CreateRdci, SetRdci

RdciRecord

This structure contains the contents of the RDCI record that is either being retrieved from the database or being created or modified.

Syntax

```
typedef struct rdci_record
{
    char document_number[21] ;
    char recvd_dci_status_code[16] ;
    char patient[11] ;
    char dci_short_name[11] ;
    char dci_name[31] ;
    char dci_date[DATE_SIZE + 1] ;
    char dci_time[TIME_SIZE + 1] ;
    char visit_name[21] ;
    int visit_number ;
    short int subevent_number ;
    char site[11] ;
    char investigator[11] ;
    DCAPIFlag blank_flag ;
    char comment[COMMENT_SIZE + 1] ;
    char received_dci_entry_ts[DATE_TIME_SIZE + 1] ;
    char entered_by[31] ;
    char end_ts[DATE_TIME_SIZE + 1] ;
    DCAPIFlag data_lock_flag ;
    char data_lock_ts[DATE_TIME_SIZE + 1] ;
    char  accessible_ts[DATE_TIME_SIZE + 1] ;
    char last_status_change_ts[DATE_TIME_SIZE + 1] ;
    char last_new_ver_ts[DATE_TIME_SIZE + 1] ;
    char modification_ts[DATE_TIME_SIZE + 1] ;
    char modified_by[31] ;
    char log_in_ts[DATE_TIME_SIZE + 1] ;
    AuditInfo   audit ;
 } RdciRecord;
```

Parameters

document_number (out)

recvd_dci_status_code (out)

patient (out)

dci_short_name (out)

dci_name (out)

dci_date (out)

dci_time (out)

visit_name (out)

visit_number (out)

subevent_number (out)

site (out)

investigator (out)

blank_flag (out)

comment (out)

received_dci_entry_ts (out)

entered_by (out)

end_ts (out)

data_lock_flag (out)

data_lock_ts (out)

accessible_ts (out)

last_status_change_ts (out)

last_new_ver_ts (out)

modification_ts (out)

modified_by (out)

log_in_ts (out)

audit (out)

Used in Functions

CreateRdci, FetchRdci, GetRdci, ProcessRdci, SetRdci

RdcmArr

This structure is used by ProcessRdci, FetchRdci, and GetRdcmArr.

Syntax

```
typedef struct rdcm_arr
{
    double rdcm_a[MAX_DCMS] ;
    int len  ;
}RdcmArr;
```

Parameters

rdcm_a (out): an array of the received_dcm_ids of all the RDCM records.

len (in): the number of elements in the array.

Used in Function

ProcessRdci, GetRdcmArr, FetchRdci

RdcmKeysRecord

This structure contains the key values of the RDCM record being created or modified by SetRdcm.

Syntax

```
typedef struct rdcm_keys_record
{
  DCAPIFlag blank_flag ;
  char qualifying_val[71] ;
  char dsp_lab[27] ;
  char visit_name[21] ;
  short int subevent_number ;
  char dcm_date[DATE_SIZE + 1] ;
  char dcm_time[TIME_SIZE + 1] ;
  char comment[COMMENT_SIZE + 1] ;
  char data_comment[COMMENT_SIZE + 1] ;
  AuditInfo   audit ;
} RdcmKeysRecord;
```

Parameters

The parameters qualifying_val, dsp_lab, visit_name, subevent_number, dcm_date, and dcm_time are only applicable if the current DCM needs them to be collected. Otherwise they must be passed in to any API function as null values. Passing a non-null value for any of these items when it is not applicable will cause the API function to fail. A null value for a character array field is the null ("") string. For integers use -1 to indicate a null value.

blank_flag (in): Value passed in is TRUE or FALSE.

qualifying_val (in):

dsp_lab (in):

visit_name (in):

subevent_number (in):

dcm_date (in):

dcm_time (in):

comment (in):

data_comment (in):

audit (in): A parameter of type AuditInfo, which tracks audit information about the RDCM changes.

Used in Function

SetRdcm

RdcmRecord

This structure contains the contents of the RDCM record that is either being retrieved from the database or being created or modified.

Syntax

```
typedef struct rdcm_record
{
    char dcm_short_name[5];
    char dcm_name[17];
    short int dcm_subset_sn;
    short int dcm_layout_sn;
    DCAPIFlag  blank_flag;
    char recvd_dcm_status_code[16];
    char qualifying_value[71];
    char dsp_lab[27];
    char visit_name[21];
    int visit_number;
    short int  subevent_number;
    char dcm_date[DATE_SIZE + 1];
    char dcm_time[TIME_SIZE + 1];
    char comment[COMMENT_SIZE + 1] ;
    char data_comment[COMMENT_SIZE + 1] ;
    char recvd_dcm_entry_ts[DATE_TIME_SIZE + 1];
    char entered_by[31];
    char end_ts[DATE_TIME_SIZE +1];
    DCAPIFlag data_lock_flag;
    char data_lock_ts[DATE_TIME_SIZE + 1];
    char accessible_ts[DATE_TIME_SIZE + 1];
    char lab_modification_ts[DATE_TIME_SIZE + 1];
    char lab_assignment_type_code[16];
    char last_status_change_ts[DATE_TIME_SIZE + 1];
    char last_data_change_ts[DATE_TIME_SIZE + 1];
    char pass1_ts[DATE_TIME_SIZE + 1];
    char pass1_by[31];
    char pass2_ts[DATE_TIME_SIZE + 1];
    char pass2_by[31];
    char modification_ts[DATE_TIME_SIZE + 1];
    char modified_by[31];
    char log_in_ts[DATE_TIME_SIZE + 1];
    AuditInfo   audit ;
} RdcmRecord;
```

Parameters

All of the following are output parameters.

dcm_short_name (out)

dcm_name (out)

dcm_subset_sn (out)

dcm_layout_sn (out)

blank_flag (out)

recvd_dcm_status_code (out)

qualifying_value (out)

dsp_lab (out)

visit_name (out)

visit_number (out)

subevent_number (out)

dcm_date (out)

dcm_time (out)

comment (out)

data_comment (out)

recvd_dcm_entry_ts (out)

entered_by (out)

end_ts (out)

data_lock_flag (out)

data_lock_ts (out)

accessible_ts (out)

lab_modification_ts (out)

lab_assignment_type_code (out)

last_status_change_ts (out)

last_data_change_ts (out)

pass1_ts (out)

pass1_by (out)

pass2_ts (out)

pass2_by (out)

modification_ts (out)

modified_by (out)

log_in_ts (out)

audit (out): A parameter of type AuditInfo, which tracks audit information about the RDCM changes

Used in Functions

GetRdcm, SetRdcm

ResponseData

This structure is used by GetResponse to retrieve the data from the database or from the buffer for the response being modified.

Syntax

```
typedef struct response_data
{
   char value_text[COMMENT_SIZE + 1];
   char data_comment[COMMENT_SIZE + 1];
}ResponseData;
```

Parameters

value_text (out)

data_comment (out)

Used in Function

GetResponse

RepeatId

This structure is used by InsertRepeat and DeleteRepeat to identify uniquely the repeat.

Syntax

```
typedef struct repeat_id
{
 double group_id;
 int repeat_sn ;
 AuditInfo   audit ;
}RepeatId;
```

Parameters

group_id (in)

repeat_sn (in)

audit (in): Not used by InsertRepeat. For DeleteRepeat, this parameter of type AuditInfo tracks audit information about the repeat being deleted.

Used in Functions

InsertRepeat, DeleteRepeat

ResponseId

This structure uniquely identifies the response.

Syntax

```
typedef struct response_id
{
    double group_id;
    double question_id;
    int repeat_sn ;
} ResponseId;
```

Parameters

Each of the following is an input parameter for the API functions below, except for WriteResponses, where all three are output parameters.

group_id

question_id

repeat_sn

Used in Functions

GetResponse, GetUnivDiscrepancy, GetManualDiscrepancy, SetDataComment, SetResponseData, SetUnivDiscrepancy, SetManualDiscrepancy, and WriteResponses

StudyRecord

This structure contains the information about the live version of the study.

Syntax

```
typedef struct study_record
{
    double clinical_study_id;
    int        clinical_study_version_id;
    DCAPIFlag      study_freeze_flag;
    DCAPIFlag      second_pass_required_flag;
    DCAPIFlag      data_entry_enabled_flag;
    DCAPIFlag      first_pass_from_login_flag;
    char      last_batch_ts[DATE_TIME_SIZE+1];
    DCAPIFlag      disc_in_browse;
    DCAPIFlag      disc_res_in_de;
    DCAPIFlag      priv_update;
    DCAPIFlag      unenroll_alert;
    DCAPIFlag      browse_acc_only;
    DCAPIFlag      dci_date_required;
    DCAPIFlag      univar_alert;
    DCAPIFlag      thesaurus_lov;
    DCAPIFlag      use_dci_book ;
    DCAPIFlag      ocl_thes_disc;
    DCAPIFlag      ocl_thes_lov;
}StudyRecord;
```

Parameters

All of these are output parameters.

clinical_study_id (out)

clinical_study_version_id (out)

study_freeze_flag (out): If TRUE, the study is frozen.

second_pass_required_flag (out): If TRUE, second pass required to make data accessible

data_entry_enabled_flag (out): If TRUE, production data entry is enabled

first_pass_from_login_flag (out): If TRUE, allows combined log-in and data entry

last_batch_ts (out)

disc_in_browse (out): If TRUE, allows creation/modification of manual discrepancies in Browse mode.

disc_res_in_de (out): If TRUE, allows discrepancy resolution in data entry.

priv_update (out): If TRUE, allows privileged update.

unenroll_alert (out): If TRUE, unenrolled patient alert is on.

browse_acc_only (out): Browse accessible data only

dci_date_required (out): This parameter is obsolete.

univar_alert (out): If TRUE, operator is alerted when univariate error occurs.

thesaurus_lov (out): If TRUE, LOVs are allowed on Thesaurus DVG-validated fields.

use_dci_book (out): If TRUE, DCI book is used as default entry mode

ocl_thes_disc (out): If TRUE, system creates discrepancies for TMS fields at data entry.

ocl_thes_lov (out): If TRUE, system allows LOVs on TMS fields at data entry.

Used in Function

SetStudyContext

ValueText

This structure contains the contents of the response.

Syntax

```
typedef struct value_txt
{
    char value_text[COMMENT_SIZE + 1];
}ValueText;
```

Parameter

value_text (in)

Used in Function

SetResponseData

7

Data Capture API Functions

The beginning of the chapter lists the Functions Grouped by Task that comprise the Oracle Clinical API. Each function is listed alphabetically by function name, and includes:

- a brief description of the function
- the **syntax** for calling it
- its input and output **parameters**
- **return values** and their meaning
- **comments**, which provide more information about the function including restrictions on its usage. Details about the task or tasks a function performs appear in this section.
- a list of **error messages** that the call can generate
- **related functions** in the DCAPI

Functions Grouped by Task

Table 7–1 groups the API functions by their task. These functions are listed alphabetically throughout the rest of this section.

All – refers to a function that can be used throughout the API development process.

Data Entry – refers to a function that processes response data.

Log-In – refers to a function that processes RDCI/RDCM data.

Standalone – refers to a function that is not related to the other functions.

Table 7–1 DCAPI Functions by Task

Task	API Function
All	ConnectOCL
	DisconnectOCL
	GetError
	GetErrorStackSize
	SetStudyContext
Data Entry	DeleteRepeat
	FlushResponses
	GetManualDiscrepancy

Table 7–1 (Cont.) DCAPI Functions by Task

Task	API Function
	GetNumRows
	GetQuestGroupId
	GetQuestionId
	GetResponse
	GetUnivDiscrepancy
	InitializeRdcmResponses
	InsertRepeat
	SetDataComment
	SetManualDiscrepancy
	SetMiscDiscrepancy
	SetResponseData
	SetUnivDiscrepancy
	WriteResponses
Log-In	CreateRdci
	DeleteRdci
	FetchRdci
	FlushRdciRdcm
	GetRdci
	GetRdcm
	GetRdcmArr
	ProcessRdci
	ProcessRdcm
	SetExternalContext
	SetRdci
	SetRdcm
	WriteRdciRdcm
Standalone	EnrollPatient
	ExecuteMultivariate
	SetActualEvent
	SetPageStatus

ConnectOCL

Connects the user to an Oracle Clinical database in either Production or Test mode.

Syntax

```
short int ConnectOCL(char *user,
    char *password,
    char *database,
    short int mode,
    double *sessionid);
```

Parameters

user (in) The OCL database user name.

password (in) The OCL database user password.

database (in) The OCL database name.

mode (in)

- Either OCL_TEST or OCL_PROD for a standalone application.
- Either 2 (Production) or 3 (Test) to share the same database session.

> **Note:** In Oracle Clinical on Forms 10g (4.6 and higher), when invoking this function to share the same database session that is used by Oracle Forms, use a mode value of 2 for Production and 3 for Test mode. When invoking the function from a standalone application (not via Oracle Forms), use a mode value of OCL_PROD for Production and OCL_TEST for Test mode.

sessionid (out) The session ID for the current session.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

None.

Error Messages

Table 7–2 Error Messages for ConnectOCL

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285700	ERR	An open database connection already exists. Use DisconnectOCL to close the connection.
297000	ERR	Null or invalid input parameters provided.

Table 7–2 (Cont.) Error Messages for ConnectOCL

Number	Severity	Message
302200	ERR	Mode should be OCL_PROD or OCL_TEST. **NOTE**: A mode value of 2 should be used for Production and 3 for Test mode when invoking the ConnectOCL DCAPI function to share the same database session.
303800	ERR	Cannot find location_code in the reference codelist for the current connection.
304900	WRN	User has been granted too many roles: exceeds the maximum allowed.
305000	WRN	Too many messages.

Related Functions

DisconnectOCL, SetStudyContext, SetExternalContext

CreateRdci

Creates a new Received DCI for a patient.

Syntax

```
short int CreateRdci(RdciKeysRecord *rdci_keys_rec,
     int rdcirdcm_mode,
     RdciRecord *rdci_rec );
```

Parameters

rdci_keys_rec (in) A structure of type RdciKeysRecord containing the key values of the API RDCI record to be created.

rdcirdcm_mode (in) The mode to be used for executing this function. It also stays in effect as the RDCIRDCM mode up to the next call of either CreateRdci or FetchRdci. Values are `INITIAL_LOGIN` or `KEY_CHANGES`.

rdci_rec (out) A structure of type RdciRecord that the function fills in with the content of the created API RDCI record.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

The function creates an RDCI record in the API RDCI buffer using the keys passed in, and returns the contents of the API RDCI record created. The component items of an API RDCI record that are not component items of rdci_keys_rec, such as all the numeric ID items, are implicitly assigned by this function. You use this function to:

1. flush the RDCIRDCM buffer.
2. validate each individual non-null item passed in rdci_keys_rec.

   All of these items must pass validation in order for the creation to take place; otherwise no creation takes place and the function returns FAILURE. The RDCI-RDCM buffer will still be empty as a result of the flush.

You use SetRdci to:

- assign values to the items of API RDCI record that are not assigned by CreateRdci.
- change the values of the items assigned by CreateRdci.

Error Messages

Table 7–3 Error Messages for CreateRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.

Table 7–3 (Cont.) Error Messages for CreateRdci

Number	Severity	Message
289800	ERR	This document already exists in the database.
290500	ERR	This site/investigator pair is invalid for study.
290600	ERR	Invalid investigator.
290700	ERR	Invalid site.
290800	ERR	Patient position is frozen.
290900	WRN	Patient not enrolled for this study.
291000	ERR	Patient position is invalid.
291200	ERR	DCI is inactive or invalid.
291300	ERR	[error message returned by OCL_CLIENT_PACK. Validate document]
291400	ERR	Clinical planned event is invalid for the current study.
294600	ERR	Document found in different study. Change study to access.
294900	WRN	Site is currently inactive.
295000	WRN	Investigator is currently inactive.
296600	ERR	Cannot preserve a lock when performing a rollback.
297000	ERR	Null or invalid input pointers provided.
297100	ERR	Changes pending.
299000	ERR	Only browse mode can be performed on a frozen study.
299700	WRN	The data entered for this received DCI will be deleted.
299800	ERR	Date cannot exceed today's date.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
305600	ERR	Invalid DCI date.
305700	ERR	Invalid DCI time.
308700	ERR	Cannot set RDCI time until RDCI short name is set.
308800	ERR	DCI short name is not applicable.
309100	ERR	Document number is not applicable.
309200	ERR	Patient is not applicable.
309400	ERR	Date is not applicable.
309600	ERR	Time is not applicable.
309800	ERR	Visit name is not applicable.
310000	ERR	Subevent number is not applicable.
310200	ERR	Site is not applicable.
310400	ERR	Investigator is not applicable.
310600	ERR	Blank flag is not applicable.
310800	ERR	Comment is not applicable.
311600	ERR	DCI document number can not be updated to null.

Table 7–3 (Cont.) Error Messages for CreateRdci

Number	Severity	Message
311700	ERR	Document number must not have any lower case.

Related Functions

SetRdci, GetRdci, ProcessRdci, FetchRdci, WriteRdciRdcm

DeleteRdci

Deletes the currently buffered RDCI record from the database along with all of its accompanying RDCMs, Responses, Discrepancies, and Actual Events (if not referenced by another RDCI or RDCM). DeleteRdci also performs a database commit and clears the API RDCIRDCM buffer.

Syntax

```
short int DeleteRdci();
```

Parameters

None.

Return Value

`SUCCESS` or `FAILURE`.

Comments

None.

Error Messages

Table 7–4 Error Messages for DeleteRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
297100	ERR	Changes pending.
306200	ERR	RDCI record being deleted does not exist in the database.
306800	ERR	Invalid audit reason.
312200	ERR	Cannot delete Received DCI as it is Data Locked.
317800	ERR	Cannot delete accessible Received DCI in Initial Login Mode.
659700	ERR	No default audit reason.
659800	ERR	More than one default audit reason found.
660000	ERR	Cannot use system-generated audit reason.

Related Functions

CreateRdci, FetchRdci, GetRdci, ProcessRdci, SetRdci,WriteRdciRdcm

DeleteRepeat

Deletes a question group repeat from the API responses buffer.

Syntax

```
short int DeleteRepeat(RepeatId*repeat_identifier );
```

Parameters

repeat_identifier (in) A parameter of type RepeatId, which contains the unique identifier for the question group repeat and the AuditInfo structure for this repeat.

Return Value

SUCCESS or FAILURE.

Comments

This function re-sequences all repeats following the one that was deleted, if any, by subtracting 1 from the repeat sequence number for all questions in the group in those repeats.

Error Messages

Table 7–5 Error Messages for DeleteRepeat

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
297100	ERR	DCM question group does not exist in the DCM.
287800	ERR	Cannot delete repeats in browse mode.
287900	ERR	Cannot delete repeat for a non-repeating question group.
288000	ERR	Repeat sn does not exist for DCM question group.
288100	ERR	Cannot delete for repeating defaults when repeat is less than the maximum repeats for DCM question group.
297000	ERR	Null or invalid input pointers provided.
302600	ERR	No existing manual discrepancy for this question.
306800	ERR	Invalid audit reason.
313600	WRN	You have privileged update and have deleted repeating default when repeat sn is less than the maximum repeats for DCM question group.
659700	ERR	No default audit reason.
659800	ERR	More than one default audit reason found.
660000	ERR	Cannot use system-generated audit reason.
685900	ERR	Audit comment for Deletion of Repeat cannot have more than 158 bytes.

Related Functions

InitializeRdcmResponses, InsertRepeat

DisconnectOCL

Ends the current Oracle Clinical database connection.

Syntax

```
short int DisconnectOCL();
```

Parameters

None.

Return Value

`SUCCESS` or `FAILURE`.

Comments

None.

Error Messages

Table 7–6 Error Messages for DisconnectOCL

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286000	ERR	Cannot disconnect while changes are pending for RDCI/RDCM work.
302300	ERR	Cannot disconnect while changes are pending for responses work.

Related Function

ConnectOCL

EnrollPatient

Enrolls a patient in the current study.

Syntax

```
short int EnrollPatient(double pat_pos_id,
    PatientRecord *pat_rec);
```

Parameters

pat_pos_id (in) The PATIENT_POSITION_ID for the patient you want to enroll. This parameter is the unique identifier in the database.

pat_rec (in) A structure of type PatientRecord.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

This function performs a database commit.

Error Messages

Table 7–7 Error Messages for EnrollPatient

Number	Severity	Message
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
288900	ERR	Cannot enroll a patient when RDCI is locked.
297000	ERR	Null or invalid input pointers provided.
302400	WRN	Zero rows being updated
303600	ERR	Invalid patient.
304800	ERR	User is not authorized to call this function.

Related Functions

None.

ExecuteMultivariate

Launches multivariate Procedure execution.

Syntax

```
short int ExecuteMultivariate(MvInfo *mv,
   DCIAPIFlag preserve_lock),
   double *return_job_id));
```

Parameters

mv (in) Criteria for multivariate validation submission. This parameter is a structure of type MvInfo.

preserve_lock (in) A flag indicating whether the lock on the API RDCI record should be preserved or released. If the value is TRUE, the lock remains; if the value is FALSE, the lock is released.

return_job_id (out) A unique identifier of the submitted job.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

This function allows the calling application to submit multivariate Procedures for execution. Execution is performed on demand for a given patient, either on a DCM basis or not, depending upon the criteria chosen for submission. Once this function is called, control is immediately returned to the application; all processing takes place in the background. As soon as the function completes its processing, the results of the execution become available in Oracle Clinical tables. For information about how to define the execution context for a Procedure, see "Procedure Classification" on page 5-6.

Results polling

From the API submission, the returned argument (return_job_id) acts as a handle for subsequent status/results polling.

Execution is tracked in a table called MV_EXECUTION_LOG, which contains the information that is listed in the following table.

Table 7–8 MV execution log information

Column Name	Type/Size	Nullability	Possible Values
JOB_ID	NUMBER	NOT NULL	
SESSION_ID	NUMBER	NOT NULL	
EXECUTION_STATUS	VARCHAR2(15)	NOT NULL	SUBMITTED, EXECUTING, COMPLETED, SUCCESS, WARNING, FAILURE
OUTCOME_STATUS	VARCHAR2(15)		

Table 7–8 (Cont.) MV execution log information

Column Name	Type/Size	Nullability	Possible Values
EXECUTION_SUBMITTED_TS	DATE	NOT NULL	
EXECUTION_START_TS	DATE		
EXECUTION_END_TS	DATE		
EXECUTION_CONTEXT	VARCHAR2(15)	NOT NULL	
PROCEDURE_ID	NUMBER(10)		
PROCEDURE_VERSION_SN	NUMBER(3)		
PATIENT_ID	NUMBER(10)	NOT NULL	
DCM_ID	NUMBER(10)		
TOT_NEW_DISC	NUMBER(10)		
TOT_SAME_DISC	NUMBER(10)		
TOT_OBS_DISC	NUMBER(10)		

The client application can:

- check EXECUTION_STATUS at any time by using the JOB_ID unique identifier.
- browse all submissions for a given user session by querying the SESSION_ID field.

When the job is submitted a corresponding row is added to the table with an:

- EXECUTION_STATUS of SUBMITTED.
- EXECUTION_SUBMITTED_TS of SYSDATE.

When DBMS_JOB starts executing the job:

- EXECUTION_STATUS is updated to EXECUTING.
- EXECUTION_START_TS is updated to SYSDATE.

Finally, upon execution completion:

- EXECUTION_STATUS changes to COMPLETED.
- OUTCOME_STATUS updates accordingly.
- The total number of new discrepancies is now available in TOT_NEW_DISC.
- The total number of unchanged discrepancies is now available in TOT_SAME_DISC.
- The total number of obsolete discrepancies is now available in TOT_OBS_DISC.
- EXECUTION_END_TS is populated with SYSDATE.

Discrepancy information can be queried from the DISCREPANCY_MANAGEMENT view which is described in the *Oracle Clinical Stable Interface Technical Reference Manual*.

Error Messages

Table 7–9 Error Messages for ExecuteMultivariate

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.

Table 7–9 (Cont.) Error Messages for ExecuteMultivariate

Number	Severity	Message
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
291000	ERR	Patient position is invalid.
297100	ERR	Changes pending.
297200	ERR	Invalid execution context.
297300	ERR	Procedure ID is required for SINGLE execution context.
297400	ERR	Procedure version is required for SINGLE execution context.
297500	ERR	Invalid Procedure (nonexistent, retired or pre-3.1 style).
297600	WRN	Procedure is not active or needs to be regenerated.
297700	WRN	Procedure is not compiled.
297800	ERR	DCM ID is required for ON-LINE/DCM execution context.
297900	ERR	DCM ID does not apply for ON-LINE execution context.
298000	ERR	DCM ID does not apply for SINGLE execution context.
298100	ERR	Procedure ID and version must be -1 for ON-LINE and ON-LINE/DCM execution contexts.
298200	ERR	Mandatory patient position ID is missing.
298300	ERR	Invalid preserve lock value.
298400	ERR	Mandatory clinical study ID is missing.
298500	ERR	Mandatory clinical study version number is missing.
298600	ERR	Invalid mode value.
298700	ERR	Invalid debug flag value.
298800	ERR	Mandatory execution context is missing.
300200	ERR	This patient is frozen.
303600	ERR	Invalid patient.
304400	WRN	No procedures qualify for execution.
304500	ERR	Cannot preserve lock as the Received DCI is not locked.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
304800	ERR	User is not authorized to call this function.
313800	ERR	User is not authorized to call ExecuteMultivariate (not granted RXCLIN_MOD privilege)

Related Functions

None.

FetchRdci

Use this function to fetch an existing header RDCI record from the database. FetchRdci pulls an RDCI record, along with all its underlying RDCM records, from the database into the API RDCIRDCM buffer. It returns the API RDCI record and an array of RECEIVED_DCM_IDs of all the API RDCM records.

Syntax

```
short int FetchRdci(double received_dci_id,
    DCIAPIFlag lock_flag,
    int rdcirdcm_mode,
    RdciRecord *rdci_rec,
    RdcmArr *rdcm_arr);
```

Parameters

received_dci_id (in) The received_dci_id of the RDCI record to be fetched. Use the stable interface to retrieve the correct ID. See the *Oracle Clinical Stable Interface Technical Reference Manual* for the documented data model.

lock_flag (in) A flag indicating whether the specified RDCI record should be locked or not. Values are true for a locked record or false for an unlocked record.

rdcirdcm_mode (in) The mode to be used for executing this function. It also stays in effect as the RDCIRDCM mode up to the next call of either CreateRdci or FetchRdci. Values are initial_log-in, first_pass_entry, key_changes, update or browse.

rdci_rec (out) A structure of type RdciRecord that the function fills in with the content of the just-populated API RDCI record.

rdcm_arr (out) A structure of type RdcmArr consisting of an array of the RECEIVED_DCM_IDs of all underlying RDCM records fetched along with the specified RDCI record, and the number of elements in that array that the function fills in.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

FetchRdci flushes the RDCIRDCM buffer, then searches the database for an RDCI record whose received_dci_id equals the parameter received_dci_id. If FetchRdci finds a record, the function:

1. fetches the record, in the specified locking mode, into the API RDCI buffer.
2. fetches all the underlying RDCM records into the API RDCM buffer.
3. returns the contents of the populated API RDCI record in the parameter rdci_rec.
4. returns the RECEIVED_DCM_IDs of all the populated API RDCM records in the array component of the parameter rdcm_arr.

However, if FetchRdci does not find an appropriate record, it fails. It also fails if it finds the record but cannot get the lock required by the parameter lock_flag. A FAILURE happening at any point during the steps above causes the function to stop with FAILURE without reversing the effect of the flush.

Error Messages

Table 7–10 Error Messages for FetchRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286300	ERR	Locking mode incompatible with data entry mode.
286600	ERR	Audit reason required and not provided.
293800	WRN	You are working with a data locked record and have privileged update.
294600	ERR	Document found in different study. Change study to access.
296600	ERR	Cannot preserve lock when performing a rollback.
296700	ERR	Unable to lock RDCI.
297000	ERR	Null or invalid input pointers provided.
297100	ERR	Changes pending.
299000	ERR	Only browse mode can be performed from a frozen study.
299200	ERR	RDCI status is incompatible with querying the record in first-pass data entry mode.
299300	ERR	RDCI is accessible. You can query only inaccessible documents in first-pass data entry mode.
299400	ERR	Operator comment in browse is disabled. You cannot fetch the document in locking mode.
299500	ERR	RDCI is inaccessible. You can query only accessible documents in browse mode.
299600	ERR	Owning location of the patient is different from that of the database.
300700	WRN	Patient is frozen. The RDCI record will behave as if it were fetched in non-locking mode.
300800	WRN	Data on this RDCI is locked. The RDCI record will behave as if it were fetched in non-locking mode.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306300	ERR	Received DCI is inactive or invalid.
306500	ERR	Unable to release lock.

Related Functions

GetRdci, SetRdci, ProcessRdci, CreateRdci, WriteRdciRdcm

FlushRdciRdcm

Flushes information from the API RDCIRDCM buffer and releases any locks on the database records that were buffered there.

Syntax

```
short int FlushRdciRdcm (DCIAPIFlag discard_changes);
```

Parameters

discard_changes (in) A flag indicating whether or not this function should discard any changes pending in the RDCIRDCM buffer prior to starting its work to flush the buffer. If the value is FALSE the function returns an error if there are changes pending.

Return Value

SUCCESS or FAILURE.

Comments

This function only affects the information in the RDCIRDCM buffer. It does not affect buffered response data. To flush the data, see FlushResponses.

Error Messages

Table 7–11 Error Messages for FlushRdciRdcm

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
299600	ERR	Cannot preserve lock when performing a rollback.
297100	ERR	Changes pending.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306500	ERR	Unable to release lock.

Related Function

WriteRdciRdcm

FlushResponses

Flushes information from the API Responses buffer, then either preserves the lock on the API RDCI record or releases it, based on the value of the parameter preserve_lock.

Syntax

```
short int FlushResponses(DCIAPIFlag discard_changes,
    DCIAPIFlag preserve_lock);
```

Parameters

discard_changes (in) A flag indicating whether this function should discard any changes pending in the Responses buffer prior to starting its work to flush the buffer. If the value is FALSE the function returns an error if there are changes pending, leaving the buffer intact.

preserve_lock (in) A flag indicating whether the lock on the API RDCI record should be preserved or released. If the value is:

TRUE – the lock remains.
FALSE – the lock on the API RDCI record is released.

Return Value

SUCCESS or FAILURE.

Comments

This function does not affect the RDCIRDCM buffer. To flush the header file, see FlushRdciRdcm.

Error Messages

Table 7–12 Error Messages for FlushResponses

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
288300	ERR	The API RDCI record is not locked.
294200	ERR	An error occurred while processing the Received DCI structure and updating the PATIENT_DM_TRACKING table.
294300	ERR	An error occurred while processing the Received DCM structure and updating the received DCI.
297100	ERR	Changes pending.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306500	ERR	Unable to release lock.

Related Functions

SetResponseData, GetResponse, WriteResponses

GetError

Returns the next error message in the internal error stack. Depending upon the error that is generated, this function reflects expected API errors or internal Oracle or API errors.

Syntax

```
short int GetError(ErrorInfo *err_info);
```

Parameters

err_info (out) – This parameter contains the information about the error in a structure of type ErrorInfo.

Return Value

SUCCESS or FAILURE.

Comments

If any unexpected internal database errors are encountered during the execution of the APIs — running out of table space, for example — the call stack and the error message encountered by the deepest routine are saved in an internal error stack. Each time GetError is called, the next entry in the stack, if any, is returned; otherwise an error is raised.

Error Messages

Table 7–13 Error Messages for GetError

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.

Related Function

GetErrorStackSize

GetErrorStackSize

Returns the depth of the error message stack. This depth data allows you to process all of the returned error information.

Syntax

```
short int GetErrorStackSize(int *size);
```

Parameters

size (out) Set to the depth of the error message stack.

Return Value

SUCCESS or FAILURE.

Comments

If any unexpected internal database errors are encountered during the execution of the APIs — running out of table space, for example — the call stack and the error message encountered by the deepest routine are saved in an internal error stack. This function returns the depth of the stack.

Error Messages

Table 7–14 Error Messages for GetErrorStackSize

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.

Related Function

GetError

GetManualDiscrepancy

Returns manual discrepancy information, if any, for the response. Manual discrepancies are also referred to as operator comments. Operator comments often consist of notes about data that appears to be incorrect even though Oracle Clinical has not flagged it for any potential errors. For example, an operator might add a comment that the handwriting for a data value must be reviewed by another person.

Syntax

```
short int GetManualDiscrepancy(ResponseId *response_identifier,
    DiscInfo *return_discrepancy_info);
```

Parameters

response_identifier (in) contains the unique identifier for this response in a data structure of type ResponseId.

return_discrepancy_info (out) contains the manual discrepancy information for this response in a structure of type DiscInfo.

Return Value

SUCCESS or FAILURE.

Comments

The response_identifier uniquely identifies the response that is being processed by the external system. Upon calling this function Oracle Clinical locates the response in the database and places the manual discrepancy information for this response into the disc_info structure.

If this is an existing document, the buffers are populated with data from the database. If some manual discrepancy has been previously added for this response, that data would be returned. Otherwise NULL values would be used to populate the output structures.

If this is a new document, and SetManualDiscrepancy has never been called for this response, NULL values would be used to populate the input variable. Otherwise, the data from the last SetManualDiscrepancy call would be returned. This applies also in the case of existing documents.

Error Messages

Table 7–15 Error Messages for GetManualDiscrepancy

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.

Table 7–15 (Cont.) Error Messages for GetManualDiscrepancy

Number	Severity	Message
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
302700	ERR	No existing manual discrepancy for the question.

Related Functions

InitializeRdcmResponses, SetManualDiscrepancy

GetNumRows

Returns the number of rows for a question group. This information might be useful if you are allocating space to hold returned values.

Syntax

```
short int GetNumRows(double dcm_group_id,
    int *num_rows);
```

Parameters

dcm_group_id (in) The question_group_id of the DCM question group.

num_rows (out) The number of rows for this DCM question group.

Return Value

SUCCESS or FAILURE.

Comments

The group_identifier uniquely identifies the question_group. Upon calling this function Oracle Clinical returns the number of rows that are present for this group in the internal buffer.

Error Messages

Table 7–16 Error Messages for GetNumRows

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
308600	ERR	Record changed by another user.

Related Functions

InitializeRdcmResponses, InsertRepeat, DeleteRepeat

GetQuestGroupId

Returns the DCM question group ID for the DCM question group. As an alternative, you can query the stable views for this information. See the *Oracle Clinical Stable Interface Technical Reference Manual* for the documented data model.

Syntax

```
short int GetQuestGroupId(GroupId *group_identifier,
    double *group_id);
```

Parameters

group_identifier (in) contains the unique name of the DCM question group in a data structure of type GroupId.

group_id (out) The internal question_group_id for the DCM question group identified by the group name.

Return Value

SUCCESS or FAILURE.

Comments

The group_identifier uniquely identifies the DCM question group. Upon calling this function Oracle Clinical locates the group in its internal structure and returns the group_id DCM question. The current state of the internal buffers — that is, the current DCM and subset_number — determines the behavior of Oracle Clinical.

Error Messages

Table 7–17 Error Messages for GetQuestGroupId

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.

Related Functions

GetResponse, GetQuestionId

GetQuestionId

Returns the DCM question_id for the DCM question. As an alternative, you can query the stable views for this information. See the *Oracle Clinical Stable Interface Technical Reference Manual* for the documented data model.

Syntax

```
short int GetQuestionId(QuestionId *question_identifier,
    double *question_id);
```

Parameters

question_identifier (in) contains the unique identifier for the question in a data structure of type QuestionId.

question_id (out) The internal dcm_question_id for the DCM question identified by the question_identifier.

Return Value

SUCCESS or FAILURE.

Comments

The question_identifier uniquely identifies the group that is being processed by the external system. Upon calling this function Oracle Clinical locates the question in its internal structure and gets the question_id. Oracle Clinical functions based on the current state of its internal buffers, that is, the current DCM and the subset_number.

Error Messages

Table 7–18 Error Messages for GetQuestionId

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.

Related Functions

GetResponse, GetQuestGroupId

GetRdci

Returns the RDCI structure from the API buffers. This function is necessary when you call WriteRdciRdcm with preserve lock. After calling WriteRdciRdcm, call GetRdci to synchronize the information on the RDCI with the information in the buffer that was updated by WriteRdciRdcm. You may then begin data entry by calling InitializeRdcmResponses.

Syntax

```
short int GetRdci(RdciRecord *rdci_rec);
```

Parameters

rdci_rec (out) contains the contents of the API RDCI record in a data structure of type RdciRecord.

Return Value

SUCCESS or FAILURE.

Comments

This function fails if the API RDCI buffer is empty. In order to:

- populate an empty API RDCI buffer, see CreateRdci and FetchRdci.
- change the contents of the API RDCI record, see SetRdci.
- process the API RDCI record as a unit, see ProcessRdci.

Error Messages

Table 7–19 Error Messages for GetRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
297000	ERR	Null or invalid input pointers provided.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.

Related Functions

SetRdci, ProcessRdci, CreateRdci, FetchRdci, WriteRdciRdcm

GetRdcm

Returns the contents of a specific API RDCM record. Typically, you call this function after calling FetchRdci for an existing document or ProcessRdci for a new document.

Syntax

```
short int GetRdcm(double received_dcm_id,
    RdcmRecord *rdcm_rec);
```

Parameters

received_dcm_id (in) The RECEIVED_DCM_ID of the API RDCM record to be returned.

rdcm_rec (out) contains the contents of the specified API RDCM record in a data structure of type RdcmRecord.

Return Value

SUCCESS or FAILURE.

Comments

This function fails if:

- the API RDCI buffer is empty.
- there are changes pending in the API RDCI buffer that have not yet been processed through a call to ProcessRdci.
- there is no API RDCM record whose RECEIVED_DCM_ID equals the parameter received_dcm_id.

In order to:

- populate an empty API RDCI buffer, see FetchRdci or CreateRdci.
- change the contents of a certain API RDCM record, see SetRdcm.
- process an API RDCM record as a unit, see ProcessRdcm.

Error Messages

Table 7–20 Error Messages for GetRdcm

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286100	ERR	RDCM with this received_dcm_id does not exist.
286600	ERR	Audit reason required and not provided.
297000	ERR	Null or invalid input pointers provided.
304600	ERR	Function does not exist.

Table 7–20 (Cont.) Error Messages for GetRdcm

Number	Severity	Message
304700	ERR	Privilege does not exist.
306100	ERR	Changes pending that have not been processed.

Related Functions

SetRdcm, ProcessRdcm, GetRdcmArr, WriteRdciRdcm

GetRdcmArr

Returns an array containing the RECEIVED_DCM_IDs of all the API RDCM records. This function is useful because it provides you with the internal keys you need to query RDCM and to look at the full record for one or more RDCMs.

Syntax

```
short int GetRdcmArr(RdcmArr *rdcm_arr);
```

Parameters

rdcm_arr (out): a structure of type RdcmArr containing an array of the RECEIVED_DCM_IDs of all API RDCM records and the number of elements in that array.

Return Value

SUCCESS or FAILURE.

Comments

This function fails if:

- the API RDCI buffer is empty.
- there are changes pending in the API RDCI buffer that have not yet been processed through a call to ProcessRdci.

In order to:

- to get the API RDCM records, see GetRdcm.
- to process an API RDCM record as a unit, see ProcessRdcm.

Error Messages

Table 7–21 Error Messages for GetRdcmArr

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
297000	ERR	Null or invalid input pointers provided.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306100	ERR	Changes pending that have not been processed.

Related Functions

GetRdcm, SetRdcm, ProcessRdcm, WriteRdciRdcm

GetResponse

Returns response value and comments as well as an indicator of whether any discrepancy information exists for the response. Also returns the response ID.

Syntax

```
short int GetResponse(ResponseId *response_identifier,
    ResponseData *response_data,
    double *response_id,
    short  int *disc_type);
```

Parameters

response_identifier (in) A ResponseId structure containing the unique identifier for this response.

response_data (out) A pointer to the ResponseData structure containing the data for this response and the data comment for this response.

response_id (out) The response_id of the response identified by a response_identifier.

disc_type (out) Indicates whether or not the response has a discrepancy. If yes, it describes the discrepancy type which may be univariate (U), manual (M), or both (B).

Return Value

SUCCESS or FAILURE.

Comments

InitializeRdcmResponses should be called before calling GetResponse to initialize the API Responses buffer with the definition of the DCM.

The response_identifier uniquely identifies the response that is being processed by the external system. Upon calling this function, Oracle Clinical locates the response in its internal structure and gets the data, comment and discrepancy information for this response into the output variables. The output variables, which are provided to hold the data, comment and discrepancy information, should be allocated by the calling routine.

If this is a new document, and SetResponseData and/or SetDataComment and/or SetUnivDiscrepancy has never been called for this response, NULL values would be used to populate the output variable. Otherwise, the data from the last SetResponseData and/or SetDataComment and/or SetUnivDiscrepancy call would be returned. This applies also in the case of existing documents, with one exception:

- if the question has a repeating default, even if SetResponseData has not been called for the response, the value of the repeating default, if provided, is returned.

For ordinary defaults, the default value is returned only for the first repeat. For repeats greater than 1, SetResponseData must be called with the appropriate default value.

> **Note:** The discrepancy information is only an indicator of whether or not any discrepancies exist for this response. To get the actual value of the discrepancy, use GetUnivDiscrepancy.

Error Messages

Table 7–22 Error Messages for GetResponse

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.

Related Functions

InitializeRdcmResponses, SetDataComment, SetResponseData, SetUnivDiscrepancy, GetUnivDiscrepancy, GetManualDiscrepancy

GetUnivDiscrepancy

Returns system-generated discrepancy information, if there is any, for the response. Use this function to query specific system-generated discrepancies.

Syntax

```
short int GetUnivDiscrepancy(ResponseId *response_identifier,
    DiscInfo *return_discrepancy_info);
```

Parameters

response_identifier (in) The ResponseId definition containing the unique identifier for this response.

return_discrepancy_info (out) The DiscInfo containing the discrepancy information for this response.

Return Value

SUCCESS or FAILURE.

Comments

The response_identifier uniquely identifies the response that is being proceeded by the external system. Upon calling this function Oracle Clinical locates the response in its internal structure and gets the discrepancy information for this response into the input variables. The input variables, which are provided for holding the discrepancy information, should be allocated by the calling routine.

For existing documents, if a discrepancy has been previously added for this response, that information would be returned. The function returns FAILURE otherwise.

For new documents, if SetResponseData and/or SetUnivDiscrepancy has never been called for this response, the function returns FAILURE because no discrepancies exist. Otherwise, the discrepancy information created by the last SetResponseData and/or SetUnivDiscrepancy call would be returned. This applies also in the case of existing documents.

Error Messages

Table 7–23 Error Messages for GetUnivDiscrepancy

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.

Table 7–23 (Cont.) Error Messages for GetUnivDiscrepancy

Number	Severity	Message
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.

Related Functions

InitializeRdcmResponses, SetResponseData, SetUnivDiscrepancy

InitializeRdcmResponses

Once you have created a record for a received DCI using CreateRdci or fetched a record using FetchRdci, you may add response data. InitializeRdcmResponses initializes the DCM, responses, and discrepancies information and then sets up the document for your chosen mode of data entry. Use this function when you want to pull up a response in the database. Calling this function sets up the buffer structure either with data or without data depending upon the information that has already been saved.

InitializeRdcmResponses allows authorized users to:

- open documents in Browse mode in order to create user comments (manual discrepancies).
- open locked documents in Update mode in order to browse for comments.

Syntax

```
short int InitializeRdcmResponses(double received_dcm_id,
    int responses_mode);
```

Parameters

received_dcm_id (in) The internal key for this document.

responses_mode (in) The mode to be used for the execution of this function. Valid values for this mode include:

- INITIAL_LOGIN
- KEY_CHANGES
- FIRST_PASS_ENTRY
- UPDATE
- BROWSE

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

InitializeRdcmResponses handles memory allocation. If you attempt to set or get response values for a question in a particular repeat in a question group without creating a repeat first, you will receive an error message. For more information about memory allocation, see "Memory Allocation Rules for Data Entry" on page 5-5.

Prior to calling this function, the RDCIRDCM buffers must be populated. The external application obtains the received_dcm_id from the stable views or from RDCI/RDCM APIs. The received_dcm_id uniquely identifies the document that is being used by the external system. Upon calling this function, Oracle Clinical initializes its internal structures with the definition of the DCM for which this document has been or will be entered and the data (if any) that had been earlier entered for this document. If this is a new document — a document with no existing data — the responses for the first repeat in all the question groups in this document are initialized to BLANK and the remaining repeats are not initialized.

An error is returned if:

- the document status is incompatible with the mode of data entry — for example, if the user attempts to update responses on a document with status RECEIVED.
- the mode of data entry is incompatible with the mode of Log-In — for example, if the user attempts in Log-In to update a document reserved in non-locking mode.
- the user does not have the adequate privileges for the attempted data entry operation.
- the study is not enabled for data entry in production mode.

Error Messages

Table 7–24 Error Messages for InitializeRdcmResponses

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286100	ERR	RDCM with this received_dcm_id does not exist.
286200	ERR	Status of Received DCM with this received_dcm_id is incompatible with data entry mode.
286300	ERR	Locking mode incompatible with data entry mode.
286500	ERR	Study is not enabled for data entry.
286600	ERR	Audit reason required and not provided.
293800	WRN	You are working with a data locked record and have privileged update.
297100	ERR	Changes pending.
298600	WRN	Invalid mode value.
299100	ERR	No data has been entered for this RDCM.
299900	ERR	Cannot proceed when blank flag is set to Y.
300100	ERR	Only the user who performed Pass 1 Entry can perform data entry task.
300200	ERR	This patient is frozen.
300300	ERR	This Received DCI is data locked.
300500	ERR	First pass entry cannot be performed on an accessible RDCM.
300600	ERR	RDCI status not compatible with the mode.
301500	ERR	Data entry is not enabled for the clinical study.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306600	ERR	This received DCM is locked.

Related Function

WriteResponses

InsertRepeat

Inserts a new record in the internal buffers for the repeat sequence number. This function is useful when entering repetitive data into a DCM. For example, you may wish to create a repeat that records time of day and patient's temperature at several different intervals within a 24-hour period. This function can be used to insert a repeat before existing repeats with resequencing of the following repeats.

Once an operator creates a blank row in the Oracle Clinical internal buffers, it is possible to add data to this new row using SetResponseData. WriteResponses is then used to commit this data into the database.

Syntax

```
short int InsertRepeat(RepeatId *repeat_identifier);
```

Parameters

repeat_identifier (in) The RepeatId containing the unique identifier for this repeat.

Return Value

SUCCESS or FAILURE.

Comments

The repeat_identifier uniquely identifies the row that is being processed by the external system (returning FAILURE if the row does not exist). Upon calling this function, Oracle Clinical inserts a new record (with all data values initialized to NULL) in its internal buffers. The API returns a FAILURE if the repeat sequence number exceeds the maximum repeats specified for the question group. SetResponseData needs to be called to update the response values as and when they are entered.

In InitializeRdcmResponses, Oracle Clinical automatically creates the first row in each group, so this function accepts repeat sequence numbers greater than or equal to 1 only.

> **Note:** This API function re-sequences all repeats, if any, following the one that was inserted by adding 1 to the repeat sequence number for all questions in the group in those repeats.

Error Messages

Table 7–25 Error Messages for InsertRepeat

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
287100	ERR	DCM question group does not exist in the DCM.

Table 7–25 (Cont.) Error Messages for InsertRepeat

Number	Severity	Message
287200	ERR	Cannot insert repeats in browse mode.
287300	ERR	Invalid DCM question group ID passed.
287400	ERR	Cannot insert repeat for a non-repeating question group.
287500	ERR	Inserting repeat will introduce a null repeat in the middle.
287600	ERR	Cannot insert for repeating defaults when repeat is less than the maximum repeats for DCM question group.
287700	ERR	The repeat exceeds the maximum repeats expected for DCM question group.
297000	ERR	Null or invalid input pointers provided.

Related Functions

InitializeRdcmResponses, SetResponseData

ProcessRdci

Purpose

Once you have completed all of the appropriate keys in the DCI, use this function to:

- validate the RDCI record as a unit.
- review interdependencies between fields.
- verify that there are no duplicate pre-existing RDCIs in the database.
- create RDCM records in the RDCM buffer for RDCMs that do not yet exist.
- perform data cascades from the API RDCI record to the existing RDCM records.

After this function verifies that the header is complete, it brings the appropriate DCMs to the buffers, and returns both the API RDCI record and an array containing the RECEIVED_DCM_IDs of all the API RDCM records.

The component items of the RDCI record that are also component items of RDCI_KEYS_REC — Investigator and Site, for example — might be implicitly assigned by this function. The component items of the RDCI record that are not component items of the RDCI_KEYS — Patient Position ID and Internal Timestamp, for example — are derived from the interface and implicitly returned by the system. For this reason the function returns the updated API RDCI record in the output parameter RDCI_REC.

Syntax

```
short int ProcessRdci(RdciRecord *rdci_rec,
    RdcmArr *rdcm_arr);
```

Parameters

rdci_rec (out) The updated contents of the API RDCI record in a RdciRecord structure.

rdcm_arr (out) A RdcmArr structure consisting of an array of the RECEIVED_DCM_IDs of all API RDCM records and the number of elements in that array.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

This function fails if:

- The API RDCI buffer is empty.
- The API RDCI record exists in the database and has not been locked.

  To learn about locking the API RDCI record see FetchRdci, WriteRdciRdcm, WriteResponses,and FlushResponses.

To validate the API RDCI record as a unit, this function:

1. checks that all the mandatory items in the API RDCI record are populated.
2. performs cross-item validation.
3. performs uniqueness validation for the API RDCI record.

In interactive applications, at any point in time, the FE is working on one buffer only; either the FE RDCI buffer or the FE RDCM buffer. If it wants to switch from working on the FE RDCI buffer to working on the FE RDCM buffer, it should follow these steps first:

1. Call ProcessRdci if there are changes pending in the API RDCI record that have not been processed through a call to ProcessRdci.
2. Refresh the FE RDCM buffer because step 1 could have resulted in changes being cascaded to the API RDCM records. Cascaded changes will include deleting the existing API RDCM records and creating new ones in case the FE has changed the DCI_NAME in the API RDCI record. In this case the FE should clear the FE RDCM buffer altogether and populate it with the new API RDCM records.

To write the contents of the API RDCIRDCM buffer to the database, see WriteRdciRdcm. WriteRdciRdcm fails if there are changes pending in the API RDCI record that have not been processed through a call to ProcessRdci.

Error Messages

Table 7–26 Error Messages for ProcessRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286100	ERR	RDCM with this received_dcm_id does not exist.
286600	ERR	Audit reason required and not provided.
289900	ERR	Document number not derived.
290000	ERR	Duplicate Received DCI found with the same keys in document number: \0.
290300	ERR	Current record is not complete: Operation was not successful.
290500	ERR	This site/investigator pair is invalid for this study.
291500	WRN	Subevent not found: actual event will be created once you save.
292000	ERR	Blank flag must be entered.
292100	ERR	Clinical planned event name must be entered.
292200	ERR	Patient must be entered.
292300	ERR	DCI name must be entered.
292400	ERR	Site is required.
292800	ERR	DCI date must be entered.
293000	ERR	Subevent number must be entered.
293600	ERR	Too many actual events correspond to this subevent #.
295100	WRN	Site/investigator pair is not current in study.
295300	ERR	An error occurred querying DVGs for an RDCM.
297000	ERR	Null or invalid input pointers provided.
304600	ERR	Function does not exist.

Table 7–26 (Cont.) Error Messages for ProcessRdci

Number	Severity	Message
304700	ERR	Privilege does not exist.
307000	ERR	Cannot call this function if you have not locked the record.
307100	ERR	Investigator is required.
307200	ERR	DCI Time is required.
307300	WRN	Lab could not be defaulted.
307600	ERR	Can not log-in data for a patient that is not associated with a book in a study with no default DCI book.
308200	ERR	The DCI book assigned to this patient is not active. Can not use a DCI book that is not active to log data in a production environment.
308500	ERR	The current DCI book is Retired and the environment is Test.
311800	WRN	Patient not assigned to this site.

Related Functions

GetRdci, SetRdci, CreateRdci, FetchRdci, WriteRdciRdcm

ProcessRdcm

Purpose

This function verifies that the selected RDCM record in the API RDCM buffer is a unit that includes all necessary data. For example, if the DCM requires a qualifying value, this function checks to see if one is supplied. In addition, this function verifies:

- interdependencies between fields.
- that there are no duplicate RDCMs in the API RDCM buffer.
- that there are no duplicate pre-existing RDCMs in the database.

When this function identifies a duplicate RDCM in the buffer, it returns the ID of that record in the double *RECEIVED_DCM_IDs parameter. When this function identifies a duplicate RDCM in the database, it returns an error and includes the document number of the duplicate record within the error message text.

Syntax

```
short int ProcessRdcm(double received_dcm_id,
    double *received_dcm_id_dup);
```

Parameters

received_dcm_id (in) The RECEIVED_DCM_ID of the API RDCM record to be validated.

received_dcm_ids (out) The RECEIVED_DCM_ID of the first duplicate API RDCM record encountered. This parameter will contain null if there is no duplicate API RDCM record encountered.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

This function fails if:

- the API RDCI buffer is empty.
- the API RDCI record exists in the database and has not been locked.

  To learn about locking the API RDCI record see FetchRdci, WriteRdciRdcm, WriteResponses,and FlushResponses. To learn about privileges, see the "Functional State Mapping Information" on page 5-12 and the "Functional State Transition Diagram" on page 5-16.
- there are changes pending in the API RDCI buffer that have not yet been processed through a call to ProcessRdci.
- there is no API RDCM record whose RECEIVED_DCM_ID equals the parameter received_dcm_id.

To validate an API RDCM record as a unit, this function:

1. checks that all the mandatory items in the API RDCM record are populated.
2. performs cross-item validation.

3. performs uniqueness validation for the API RDCM record. If the function finds a duplicate within the other API RDCM records, it returns its RECEIVED_DCM_ID in the output parameter RECEIVED_DCM_IDs along with an error message. However, if the function finds a duplicate under another RDCI in the database, it returns an error message.

WriteRdciRdcm also validates each of the API RDCM records as part of its processing. This function is provided in order to give the FE a facility to determine early in the process whether individual API RDCM records are valid.

To write the contents of the API RDCIRDCM buffer to the database, see WriteRdciRdcm.

Error Messages

Table 7–27 Error Messages for ProcessRdcm

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
290100	ERR	Duplicate Received DCM found within this Received DCI.
290200	ERR	Duplicate Received DCM found in document number: \0.
290300	ERR	Current record is not complete: Operation was not successful.
291500	WRN	Subevent not found: actual event will be created once you save.
292000	ERR	Blank flag must be entered.
292100	ERR	Clinical planned event name must be entered.
293000	ERR	Subevent number must be entered.
293300	ERR	DCM time must be entered.
293400	ERR	Qualifying value must be entered.
293600	ERR	Too many actual events correspond to this subevent #.
293700	ERR	DCM date must be entered.
297000	ERR	Null or invalid input pointers provided.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306100	ERR	Changes pending that have not been processed.
307000	ERR	Cannot call this function if you have not locked the record.
307300	WRN	Lab could not be defaulted.
307400	ERR	Please use the patient enrollment form to specify which DCI Book to use to maintain the page tracking information.

Related Functions

SetRdcm, GetRdcm, GetRdcmArr, WriteRdciRdcm

SetActualEvent

Use this function to add a comment to an actual event. This function updates the specified actual event record with the new comment and commits the change.

Syntax

```
short int SetActualEvent(double actual_event_id,
    ActualEventsRecord *actual_event);
```

Parameters

actual_event_id (in) The ACTUAL_EVENT_ID of the row you want to update.

ActualEventsRecord (in) The new contents with which you want to update the specified Actual Event record. This parameter is in a structure of type ActualEventsRecord.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

None.

Error Messages

Table 7–28 Error Messages for SetActualEvent

Number	Severity	Message
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
288800	ERR	Cannot set actual event when RDCI is locked.
297000	ERR	Null or invalid input pointers provided.
302400	WRN	Zero rows updated.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
304800	ERR	User is not authorized to call this function.

Related Functions

None.

SetDataComment

Use this function to modify a comment. This function updates the investigator comment, if any, for the response in the Oracle Clinical internal structure.

Syntax

```
short int SetDataComment(ResponseId *response_identifier;
    DataComment *data_comm);
```

Parameters

response_identifier (in) The ResponseId definition containing the unique identifier for this response.

data_comm (in) A DataComment structure containing the data (investigator) comment for this response.

Return Value

SUCCESS or FAILURE.

Comments

The response_identifier uniquely identifies the response that is being processed by the external system. Upon calling SetDataComment, Oracle Clinical locates the response in its internal structure and updates the data (investigator) comment information for this response from the input variables provided for holding this information. These input variables should be allocated and initialized before calling the routine.

Error Messages

Table 7–29 Error Messages for SetDataComment

Number	Severity	Message
284800	ERR	Function not available in this mode.
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
304600	ERR	Function does not exist.

Table 7–29 (Cont.) Error Messages for SetDataComment

Number	Severity	Message
304700	ERR	Privilege does not exist.
306800	ERR	Invalid audit reason.
659700	ERR	No default audit reason.
659800	ERR	More than one default audit reason found.
660000	ERR	Cannot use system-generated audit reason.

Related Functions

InitializeRdcmResponses, GetResponse

SetExternalContext

Sets the external context information that will be used to populate the columns of the RDCI_HISTORY records to be created. This function helps to create a comprehensive tracking record that becomes part of the RDCI history. For example, this function enables you to record when data was originally entered into an external system that later passes data to Oracle Clinical.

Syntax

```
short int SetExternalContext(ExternalContextInfo *context_info);
```

Parameters

context_info (in) A structure of type ExternalContextInfo.

Return Value

SUCCESS or FAILURE.

Comments

This function validates the value of the parameter context_info. It validates the trans_type item by checking that it contains a value from the reference codelist EXTERNAL_TRANS_TYPE. If it does not contain a value, the value STANDARD will be used. If the validation succeeds, the value of context_info will immediately take effect as the new external context info and the function returns SUCCESS; otherwise, the function will return FAILURE leaving the existing external context info intact.

The values set through a call to SetExternalContext will be reflected in all the subsequently created RDCI_HISTORY records until the next call of SetExternalContext or the call of DisconnectOCL, whichever comes first. No auditing records will be created if SetExternalContext has never been called since the last call to ConnectOCL.

An RDCI record in the API RDCI buffer will have only one audit record created for it upon the first writing to the database done due to a call of either WriteRdciRdcm or WriteResponses. Successive writing to the database while the same RDCI record is still in the API RDCI buffer will cause more audit records to be created for it only if a call to SetExternalContext has taken place after the last time an audit record was created for it.

In contrast to the API, Oracle Clinical does not maintain the RDCI_HISTORY table in the sense that it does not create or update records of that table.

The RDCI_HISTORY table will not be replicated and there are no reports running against it.

Error Messages

Table 7–30 Error Messages for SetExternalContext

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.

Table 7–30 (Cont.) Error Messages for SetExternalContext

Number	Severity	Message
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
297000	ERR	Null or invalid input pointers provided.
297100	ERR	Changes pending.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.

Related Functions

ConnectOCL, DisconnectOCL

SetManualDiscrepancy

Creates or updates manual discrepancy (operator comment) information for the response in the Oracle Clinical internal structure. This information includes the original comment and information about whether or not the issue has been resolved. If the issue has been resolved, the resolution is described. If unresolved, the new comment replaces the existing comment (which is retained in the DISCREPANCY_ENTRY_REVIEW_HIST table).

Syntax

```
short int SetManualDiscrepancy(ResponseId *response_identifier,
    DiscInfo *discrepancy_info);
```

Parameters

response_identifier (in) The ResponseId definition containing the unique identifier for this response.

discrepancy_info (in) The DiscInfo definition containing the discrepancy information for this response.

Return Value

SUCCESS or FAILURE.

Comments

The response_identifier parameter uniquely identifies the response that is being processed by the external system. Upon calling SetManualDiscrepancy, Oracle Clinical locates the response in its internal structure and updates the manual discrepancy (operator comment) information for this response from the input variables provided for holding this information. These input variables should be allocated and initialized before calling the routine.

If the user has the privilege to create operator comments in browse mode and the document was fetched in locked mode, SetManualDiscrepancy may be invoked even in browse mode. In this respect it differs from other functions that set response and/or discrepancy values which may not be called in browse mode.

Error Messages

Table 7–31 Error Messages for SetManualDiscrepancy

Number	Severity	Message
284800	ERR	Function not available in this mode.
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.

Table 7–31 (Cont.) Error Messages for SetManualDiscrepancy

Number	Severity	Message
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
297000	ERR	Null or invalid input pointers provided.
302800	ERR	Invalid discrepancy review status code.
302900	ERR	Invalid discrepancy resolution type code.
303000	ERR	Resolution type code cannot be null for RESOLVED review status.
303100	ERR	Resolution type code should be null if review status is not RESOLVED.
312500	ERR	Can not update repeating defaults when repeat sn is less than the maximum repeats expected for DCM question group.

Related Functions

InitializeRdcmResponses, GetManualDiscrepancy

SetMiscDiscrepancy

Sets the resolution text for an existing discrepancy with a current status. This function can only be called in UPDATE mode.

Syntax

```
short int SetMiscDiscrepancy(long disc_id, char *disc_subtype, DiscInfo
*discrepancy_info);
```

Parameters

disc_id (in) The discrepancy ID for the discrepancy being updated.

disc_subtype (in) Type of discrepancy. Valid types are UNIVARIATE, MANUAL, INDICATOR, MULTIVARIATE or MANUAL HEADER.

discrepancy_info (in) The DiscInfo definition containing the discrepancy information. Only the resolution field in the structure DiscInfo is used.

Return Value

SUCCESS or FAILURE.

Comments

The disc_id and subtype parameters uniquely identify the discrepancy that is being processed by the external system. Upon calling SetMiscDiscrepancy, Oracle Clinical locates the discrepancy in its internal structure and updates the resolution comment or text for this discrepancy from the input variables provided for holding this information. These input variables should be allocated and initialized before calling the routine. The remaining fields should be initialized, otherwise a warning is raised.

If any of the remaining fields contain a non-null value, then a warning is raised.

Only discrepancies in status 'CURRENT' can be updated using this routine.

Error Messages

Table 7–32 Error Messages for SetMiscDiscrepancy

Number	Severity	Message
992200	WRN	All other parameters except the resolution text in the discrepancy structure will be ignored.
992300	ERR	Unable to get existing discrepancy information from the database.

Related Functions

InitializeRdcmResponses

SetPageStatus

This function sets the status of a received page and commits. SetPageStatus enables you to track a specific page within a DCI. This is particularly useful, for example, when a DCI has multiple repeats within the same group that are listed on separate pages of the Case Report Form (CRF). Recording the page number from the original paper form provides a useful tracking mechanism for the electronic version of the document.

Syntax

```
short int SetPageStatus(double received_page_id,
    char *page_status);
```

Parameters

received_page_id (in) The RECEIVED_PAGE_ID of the received page whose status is to be updated.

page_status (in) The new status of the specified received page.

Return Value

SUCCESS or FAILURE.

Comments

This function will first lock the RDCI record to which the specified page belongs. If it cannot, it will fail; otherwise it will validate the new status and update the received page with the new status.

Error Messages

Table 7–33 Error Messages for SetPageStatus

Number	Severity	Message
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
289000	ERR	Cannot set page status when RDCI is locked.
289100	ERR	Unable to find the RDCI to which page belongs: \0.
289200	ERR	Unable to lock the RDCI to which page belongs.
289300	ERR	'Unknown' page status can only be changed to 'Missing' or 'Present'.
289400	ERR	The page status cannot be manually changed to a system status.
289500	ERR	A system page status cannot be manually changed.
297000	ERR	Null or invalid input pointers provided.
302500	ERR	Invalid page status.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
304800	ERR	User is not authorized to call this function.

Related Functions

None.

SetRdci

Assigns the values of the parameter rdci_keys_rec, which is the parameter for the keys within the RDCI record, to the API RDCI record. It also returns the contents of the updated API RDCI record in the output parameter rdci_rec, which is the parameter for the entire RDCI record.

The component items of an API RDCI record which are not component items of rdci_keys_rec, such as all the numeric ID items, are implicitly assigned by this function. For example, patient ID is not a key field, but because it is part of the RDCI record, it is populated implicitly once the external patient identifier is supplied and validated.

Syntax

```
short int SetRdci(RdciKeysRecord *rdci_keys_rec,
    RdciRecord *rdci_rec);
```

Parameters

rdci_keys_rec (in): The RdciKeysRecord definition containing the value that you want to assign to the API RDCI record.

rdci_rec (out): The RdciRecord definition containing the contents of the updated API RDCI record.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

This function fails if:

- the API RDCI buffer is empty.
- the API RDCI record exists in the database and has not been locked, or its locked but the user does not have the privilege required for SetPageStatus. To learn about locking the API RDCI record see FetchRdci, WriteRdciRdcm, WriteResponses and FlushResponses. To learn about privileges, see the "Functional State Mapping Information" on page 5-12 and the "Functional State Transition Diagram" on page 5-16.

Only the items of rdci_keys_rec that have a value different from that of the corresponding item in the API RDCI record will be considered. These items will be validated first. All of them have to pass validation in order for the update to take place; otherwise no update will take place and the function will return FAILURE. The type of validation taking place here is individual item validation whereas cross-item validation will happen as part of the validation of the API RDCI record as a unit which takes place in ProcessRdci.

If this function fails, rdci_rec will be empty; if it succeeds rdci_rec will contain the current contents of the updated API RDCI record.

To populate an empty API RDCI buffer, see CreateRdci and FetchRdci.

Error Messages

Table 7–34 Error Messages for SetRdci

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
289800	ERR	This document already exists in the database.
290500	ERR	This site/investigator pair is invalid for study.
290600	ERR	Invalid investigator.
290700	ERR	Invalid site.
290800	ERR	Patient position is frozen.
290900	ERR	Patient not enrolled for this study.
291000	ERR	Patient position is invalid.
291200	ERR	DCI is inactive or invalid.
291300	ERR	[error message returned by OCL_CLIENT_PACK. Validate document.]
291400	ERR	Clinical planned event is invalid for the current study.
294600	ERR	Document found in different study. Change study to access.
294900	WRN	Site is currently inactive.
295000	WRN	Investigator is currently inactive.
297000	ERR	Null or invalid input pointers provided.
299700	WRN	The data entered for this received DCI will be deleted.
299800	ERR	Date cannot exceed today's date.
300900	ERR	No update allowed to the RDCI buffer.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
305600	ERR	Invalid DCI date.
305700	ERR	Invalid DCI time.
306800	ERR	Invalid audit reason.
307000	ERR	Cannot call this function if you have not locked the record.
308700	ERR	Cannot set RDCI time until RDCI short name is set.
308900	ERR	DCI short name is not updateable.
309000	ERR	Document number is not updateable.
309300	ERR	Patient is not updateable.
309500	ERR	Date is not updateable.
309700	ERR	Time is not updateable.
309900	ERR	Visit name is not updateable.

Table 7–34 (Cont.) Error Messages for SetRdci

Number	Severity	Message
310100	ERR	Subevent number is not updateable.
310300	ERR	Site is not updateable.
310500	ERR	Investigator is not updateable.
310700	ERR	Blank flag is not updateable.
310900	ERR	Comment is not updateable.
311600	ERR	DCI comment number can not be updated to null.
311700	ERR	Document number must not have lower case.
314800	ERR	Received DCI is accessible; please use Key Changes to change Blank Flag.
315000	ERR	Some received DCMs are accessible; please use Key Changes to change Blank Flag.
315100	WRN	Responses exist for this RDCI. Changing the Blank Flag to 'Y' will cause them to be irrevocably deleted.
659700	ERR	No default audit reason.
659800	ERR	More than one default audit reason found.
660000	ERR	Cannot use system-generated audit reason.

Related Functions

GetRdci, ProcessRdci, CreateRdci, FetchRdci, WriteRdciRdcm

SetRdcm

Assigns the value of the parameter rdcm_keys_rec to a specific API RDCM record. It also returns the contents of the updated API RDCM record in the parameter rdcm_rec. The component items of API RDCM record which are not component items of rdcm_keys_rec, such as all the numeric ID items, are implicitly assigned by this function. Patient ID, for example, is not a key field but instead is populated implicitly once the external patient identifier is supplied and validated. This process is the reason that the function returns the updated API RDCM record in the output parameter rdcm_rec.

Syntax

```
short int SetRdcm(double received_dcm_id,
    RdcmKeysRecord *rdcm_keys_rec,
    RdcmRecord *rdcm_rec);
```

Parameters

received_dcm_id (in) The RECEIVED_DCM_ID of the API RDCM record to be updated.

rdcm_keys_rec (in) The RdcmKeysRecord definition containing the value that you want to assign to the specified API RDCM record.

rdcm_rec (out) The RdcmRecord definition containing the contents of the updated API RDCM record.

Return Value

SUCCESS or FAILURE.

Comments

This function fails if:

- the API RDCI buffer is empty.
- the API RDCI record exists in the database and has not been locked, or it is locked but the user does not have the privilege required for SetRdcm.

  To learn about locking the API RDCI record, see GetRdci, FetchRdci, FlushRdciRdcm, FlushResponses, WriteRdciRdcm, and WriteResponses. To learn about privileges, see the "Functional State Mapping Information" on page 5-12 and the "Functional State Transition Diagram" on page 5-16.
- there are changes pending in the API RDCI buffer that have not yet been processed through a call to ProcessRdci.
- there is no API RDCM record whose RECEIVED_DCM_ID equals the parameter received_dcm_id.

Only the items of rdcm_keys_rec that have a value different from that of the corresponding item in the specified API RDCM record will be considered. These items will be validated first. All of them have to pass validation in order for the update to take place; otherwise no update will take place and the function will return FAILURE. The type of validation taking place in SetRdcm is individual item validation, whereas cross-item validation will happen as part of the validation of the API RDCM record as a unit. Cross-item validation takes place in both ProcessRdcm and WriteRdciRdcm.

If the API RDCM record contains items related to DCM definitions, such as dcm_date and dcm_time, and those items take their values, by definition, from the corresponding items in the API RDCI record, then the values passed in for these items in SetRdcm should be the same as the values of these items in the API RDCM record. If this is not the case, the function will fail. This restriction does not apply, however, if those items do not take their values from the API RDCI record. The current values of an API RDCI record can be obtained through a call to GetRdcm.

If this function fails, rdcm_rec is empty, and if it succeeds rdcm_rec contains the current contents of the updated API RDCM record.

To populate an empty API RDCI buffer, see CreateRdci and FetchRdci.

Error Messages

Table 7–35 Error Messages for SetRdcm

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286100	ERR	RDCM with this received_dcm_id does not exist.
286600	ERR	Audit reason required and not provided.
291400	ERR	Clinical planned event is invalid for the current study.
294600	ERR	Invalid lab.
291700	ERR	Discrete value does not exist or is inactive.
291800	ERR	Discrete value could not be validated.
291900	ERR	Value must be Y or N.
293500	ERR	Cannot set to N (not blank) when Received DCI is blank.
297000	ERR	Null or invalid input pointers provided.
299800	ERR	Date cannot exceed today's date.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
305800	ERR	Invalid DCM date.
305900	ERR	Invalid DCM time.
306100	ERR	Changes pending that have not been processed.
306800	ERR	Invalid audit reason.
306900	ERR	No update allowed to the RDCM buffer.
307000	ERR	Cannot call this function if you have not locked the record.
309400	ERR	Date is not applicable.
309500	ERR	Date is not updateable.
309600	ERR	Time is not applicable.
309700	ERR	Time is not updateable.
309800	ERR	Visit name is not applicable.

Table 7–35 (Cont.) Error Messages for SetRdcm

Number	Severity	Message
309900	ERR	Visit name is not updateable.
310000	ERR	Subevent number is not applicable.
310100	ERR	Subevent number is not updateable.
310600	ERR	Blank flag is not applicable.
310700	ERR	Blank flag is not updateable.
310800	ERR	Comment is not applicable.
310900	ERR	Comment is not updateable.
311000	ERR	Qualifying value is not applicable.
311100	ERR	Qualifying value is not updateable.
311200	ERR	Lab is not applicable.
311300	ERR	Lab is not updateable.
311400	ERR	Data comment is not applicable.
311500	ERR	Data comment is not updateable.
314900	ERR	Received DCM is accessible; please use Key Changes to change Blank Flag.
315200	WRN	Responses exist for this RDCM. Changing the Blank Flag to 'Y' will cause them to be irrevocably deleted.
345400	ERR	Qualifying value can only be changed to the default value defined at the DCI module.
659700	ERR	No default audit reason.
659800	ERR	More than one default audit reason found.
660000	ERR	Cannot use system-generated audit reason.

Related Functions

GetRdcm, GetRdcmArr, WriteRdciRdcm

SetResponseData

Updates data and audit information, if any, for the response in the Oracle Clinical internal structure and returns any univariate discrepancy raised.

Syntax

```
short int SetResponseData(ResponseId *response_identifier,
    ValueText *value,
    AuditInfo *audit_comment,
    DiscInfo *return_discrepancy_info,
    DCIAPIFlag *needs_audit);
```

Parameters

response_identifier (in) The ResponseId definition containing the unique identifier for this response.

value (in) A pointer to the ValueText structure containing the data for this response.

audit_comment (in) An AuditInfo structure holding the audit comment for this response.

return_discrepancy_info (out) A DiscInfo structure containing the discrepancy information for this response.

needs_audit (out) Indicates whether the response requires audit comment.

Return Value

SUCCESS or FAILURE or WARNING.

Comments

The response_identifier uniquely identifies the response that is being processed by the external system. Upon calling SetResponseData, Oracle Clinical locates the response in its internal structure and updates the audit comment information for this response from the input variables provided for holding this information. These input variables should be allocated and initialized before calling the routine.

While updating the data, the function checks the data for a univariate discrepancy and returns any discrepancy found in the structure provided for that purpose. If updating the data clears existing discrepancies, the discrepancy buffer is also cleared.

If this document is accessible and the API is in UPDATE mode, the audit comment is mandatory. SetResponseData returns FAILURE if the audit comment is needed but not provided — however, the internal buffers are updated with the rest of the information provided and the needs_audit flag is set. The user must call SetResponseData again with the missing audit comment information before calling any other API functions which will return FAILURE if invoked. Once the response has been updated with the audit comment information, you may continue with further processing by invoking other API functions.

If the RDCM is accessible, then the audit comment is mandatory to change the data. Otherwise, the API function will return FAILURE if the audit comment is not supplied.

Note: It is possible to enter response values for non-displayed, non-enterable and derived questions. The SetResponseData API does not check for such properties of the question and leaves control with the external system.

Error Messages

Table 7–36 Error Messages for SetResponseData

Number	Severity	Message
284800	ERR	Function not available in this mode.
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306800	ERR	Invalid audit reason.
312500	ERR	Can not update repeating defaults when repeat sn is less than the maximum repeats expected for DCM question group.

Related Functions

InitializeRdcmResponses, GetResponse, GetUnivDiscrepancy

SetStudyContext

This function sets up the environment for a study by gathering all study-related information, user-related information and configuration settings. Calling this function sets the study context for future API calls.

Syntax

```
short int SetStudyContext(char *study,
    StudyRecord *study_rec);
```

Parameters

study (in) The study name.

study_rec (out) A StudyRecord structure containing information about the live version of the study.

Return Value

SUCCESS or FAILURE.

Comments

This function takes these processing steps:

1. Set the new study context, using the parameter study, replacing the existing study context. If it fails in setting the study context it will return FAILURE, leaving the existing study context intact.
2. Flush the API RDCIRDCM buffer and consequently the Responses buffer.

Error Messages

Table 7–37 Error Messages for SetStudyContext

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286500	ERR	Study is not enabled for data entry.
301300	ERR	Invalid clinical study.
301400	ERR	The clinical study is not live.
301500	ERR	Data entry is not enabled for the clinical study.
301600	ERR	Clinical study does not exist in the current location.
301700	ERR	No study state for the clinical study.
301800	ERR	No study access account exists for the study.
301900	ERR	Record validation failed.
302000	ERR	User is not authorized to access the study.
302100	ERR	Validation of study failure.

Table 7–37 (Cont.) Error Messages for SetStudyContext

Number	Severity	Message
303300	ERR	Changes are pending for RDCI/RDCM work.
303400	ERR	Changes are pending for responses work.

Related Functions

ConnectOCL, DisconnectOCL

SetUnivDiscrepancy

Updates discrepancy information such as review status and comment for the response in the Oracle Clinical internal structure.

Syntax

```
short int SetUnivDiscrepancy(ResponseId *response_identifier,
    DiscInfo *discrepancy_info);
```

Parameters

response_identifier (in) The ResponseId structure definition containing the unique identifier for this response.

discrepancy_info (in) The DiscInfo structure definition containing the discrepancy information for this response.

Return Value

SUCCESS or FAILURE.

Comments

The response_identifier uniquely identifies the response that is being processed by the external system. Upon calling SetUnivDiscrepancy, Oracle Clinical locates the response in its internal structure and updates the discrepancy information for this response from the input variables provided for holding this information. These input variables should be allocated and initialized before calling the routine.

Error Messages

Table 7–38 Error Messages for SetUnivDiscrepancy

Number	Severity	Message
284800	ERR	Function not available in this mode.
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
286700	ERR	Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.
286800	ERR	No response in the internal buffers for the question.
286900	ERR	No existing univariate discrepancy for the question.
287000	ERR	DCM question and DCM question group combination does not exist in DCM.
287100	ERR	DCM question group does not exist in DCM.
288000	ERR	Repeat sn does not exist for DCM question group.
297000	ERR	Null or invalid input pointers provided.

Table 7–38 (Cont.) Error Messages for SetUnivDiscrepancy

Number	Severity	Message
302800	ERR	Invalid discrepancy review status code.
302900	ERR	Invalid discrepancy resolution type code.
303000	ERR	Resolution type code cannot be null for RESOLVED review status.
303100	ERR	Resolution type code should be null if review status is not RESOLVED.
312500	ERR	Can not update repeating defaults when repeat sn is less than the maximum repeats expected for DCM question group.

Related Functions

InitializeRdcmResponses, GetResponse, GetUnivDiscrepancy

WriteRdciRdcm

This function reviews the header information in the API RDCM and RDCI buffers and validates that each of the API RDCM records is a unit that includes all necessary data. In addition, it verifies that there are no duplicate RDCMs in either buffer or in the database. If there are duplicate RDCMs in the buffer:

- the RDCM ID of the RDCM being tested is returned in the double *received_dcm_id parameter
- the ID of the duplicate RDCM is returned in the double *received_dcm_id_dup parameter

If there are duplicate pre-existing RDCMs in the database, an error is returned and the document number of the duplicate record is returned as part of the error message text. Once the API RDCM records are validated as a unit and all duplicates are resolved, this function writes the contents of the API RDCIRDCM buffers to the database and commits. If someone has created a duplicate RDCI between the time you called ProcessRdci and the time you called WriteRdciRdcm, the duplicate will be trapped at the time of the commit and the function will fail. Then it either preserves the lock on the API RDCI record or releases it based on the value of the parameter preserve_lock.

Syntax

```
short int WriteRdciRdcm (DCIAPIFlag preserve_lock,
    double *received_dcm_id,
    double *received_dcm_id_dup);
```

Parameters

preserve_lock (in) A flag indicating whether the lock on the API RDCI record should be preserved or released. If the value is TRUE, the lock will be kept and if it is FALSE, it will be released.

received_dcm_id (out) The RECEIVED_DCM_ID of either the first API RDCM record to fail validation in steps 1 or 2 below or the first API RDCM record in a pair of duplicate API RDCM records found in step 3 below.

received_dcm_id_dup (out) The RECEIVED_DCM_ID of the second API RDCM record in a pair of duplicate API RDCM records. This parameter will contain null if there are no duplicate API RDCM records encountered.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

To validate each API RDCM record as a unit, this function takes these steps for each API RDCM record:

1. The function checks that all the mandatory items in the API RDCM record are populated.
2. WriteRdciRdcm performs cross-item validation. If either the previous step or this step fails, the function stops with FAILURE returning in the output parameter received_dcm_id the RECEIVED_DCM_ID of the API RDCM record and null in the parameter received_dcm_id_dup.

3. The function performs uniqueness validation for the API RDCM record. If it finds a duplicate within the other API RDCM records it returns the following values:

   - In the output parameter received_dcm_id, the function returns the RECEIVED_DCM_ID of the API RDCM record currently being validated.
   - In the output parameter received_dcm_id_dup, the function returns the RECEIVED_DCM_ID of the duplicate API RDCM record found, in addition to an error message.

   If, on the other hand, WriteRdciRdcm finds a duplicate under another RDCI in the database, the function just returns an error message.

As part of the validation it carries out, this function will change the value of the blank flag of the API RDCI record to bring it in sync with the set of blank flag values of all the API RDCM records, if this change is necessary.

This function also assigns values to the time stamp items of both the API RDCI record and the API RDCM records.

To see the effects of this function on the API RDCIRDCM buffer, the front end has to call GetRdci and GetRdcm.

To validate a single API RDCM record, see ProcessRdcm.

Error Messages

Table 7–39 Error Messages for WriteRdciRdcm

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
290100	ERR	Duplicate Received DCM found within this Received DCI.
290200	ERR	Duplicate Received DCM found within document number: \0.
290300	ERR	Current record is not complete: Operation was not successful.
291500	WRN	Subevent not found: actual event will be created once you save.
292000	ERR	Blank flag must be entered.
292100	ERR	Clinical planned event name must be entered.
293000	ERR	Subevent number must be entered.
293300	ERR	DCM time must be entered.
293400	ERR	Qualifying value must be entered.
293600	ERR	Too many actual events correspond to this subevent #.
293700	ERR	DCM date must be entered.
293800	WRN	You are working with a data locked record and have privileged update.
294200	ERR	An error occurred while processing the Received DCI structure and updating the PATIENT_DM_TRACKING table.
294300	ERR	An error occurred while processing the Received DCM structure and updating the received DCI.

Table 7–39 (Cont.) Error Messages for WriteRdciRdcm

Number	Severity	Message
294700	ERR	Received DCI is protected because at least one of its received DCMs is locked.
294800	ERR	An error occurred while in the Rxc_Login_Pack. SoftDeleteAll received DCMs function.
295200	ERR	Too many rows found in NON_EXPECTED_DCMS table.
295800	ERR	Problem retrieving from the actual event sequence.
296700	ERR	Unable to lock RDCI.
297000	ERR	Null or invalid input pointers provided.
300700	WRN	Patient is frozen. The RDCI record will behave as if it were fetched in non-locking mode.
301200	WRN	No changes to save.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306100	ERR	Changes pending that have not been processed.
306400	ERR	Too many current actual events.
306500	ERR	Unable to release lock.
307000	ERR	Cannot call this function if you have not locked the record.
307400	ERR	Please use the patient enrollment form to specify which DCI Book to use to maintain the page tracking information.
315500	WRN	RDCI is not protected because at least one of its RDCMs is locked and you have privileged update.

Related Functions

CreateRdci, FetchRdci, SetRdci, GetRdci, ProcessRdci

WriteResponses

This function reviews the response data in the buffers, verifies that they are clean, and then sends it to the Oracle Clinical database. If this function detects an incomplete flag (used, for example, if the operator has not finished the work and will complete it after lunch), the response data remains in the buffers. Otherwise the function writes the data to the database and commits. Then it either preserves the lock on the API RDCI record or releases it based on the value of the parameter preserve_lock.

Syntax

```
short int WriteResponses(DCIAPIFlag incomplete,
    DCIAPIFlag preserve_lock,
    ResponseId *response_identifier);
```

Parameters

incomplete (in) – Specify whether responses should be saved in incomplete mode. Value is TRUE or FALSE.

preserve_lock (in) – A flag indicating whether the lock on the API RDCI record should be preserved or released. If the value is TRUE, the lock will be kept and if it is FALSE, it will be released.

response_identifier (out) – A ResponseId structure containing the unique identifier of the failed response.

Return Value

`SUCCESS` or `FAILURE` or `WARNING`.

Comments

None.

Error Messages

Table 7–40 Error Messages for WriteResponses

Number	Severity	Message
284900	ERR	User does not have the privilege to call function in the current mode.
285000	ERR	Function called out of sequence.
285900	ERR	No current OCL database connection open. Use ConnectOCL to connect to OCL database.
286600	ERR	Audit reason required and not provided.
288300	ERR	The API RDCI record is not locked.
288400	ERR	The completion flag is not compatible with the data entry mode.
288500	WRN	Write responses called but no changes are pending.
288600	ERR	Blank repeats exist.
288700	ERR	Mandatory field missing.
293800	WRN	You are working with a data locked record and have privileged update.

Table 7–40 (Cont.) Error Messages for WriteResponses

Number	Severity	Message
294200	ERR	An error occurred while processing the Received DCI structure and updating the PATIENT_DM_TRACKING table.
294300	ERR	An error occurred while processing the Received DCM structure and updating the Received DCI.
300700	WRN	Patient is frozen. The RDCI record will behave as if it were fetched in non-locking mode.
303500	ERR	Document contains empty repeats.
304600	ERR	Function does not exist.
304700	ERR	Privilege does not exist.
306400	ERR	Too many current actual events.
306500	ERR	Unable to release lock.
308600	ERR	Record changed by another user.

Related Functions

SetResponseData, GetResponse, FlushResponses

A

Error Messages

This section lists the error messages that may be generated when a function is called. To retrieve error messages, call the GetError function.

-1: [Text varies, according to the particular error]

Severity: ERR

Cause: Internal Oracle Clinical error.

Action: Roll back immediately. Data may have been corrupted.

284800: Function not applicable in this mode

Severity: ERR

Cause: For example, attempted to SetResponseData in Browse mode.

Action: Change to the appropriate mode in order to call the function.

284900: User does not have the privilege to call function in the current mode

Severity: ERR

Cause: The user is not authorized to access the function that is being called.

Action: Look at the table that maps function role to action and change the user's privileges.

285000: Function called out of sequence.

Severity: ERR

Cause: You are not in the right state.

Action: Refer to the "Functional State Transition Diagram" to identify the correct state for the function that you are calling.

285700: An open database connection already exists. Use DisconnectOCL to close the connection.

Severity: ERR

Cause: You are already connected to Oracle Clinical. You cannot have two sessions running concurrently.

Action: Disconnect the current session.

285900: No current OCL database connection open. Use ConnectOCL to connect to OCL database.

Severity: ERR

Cause: There is no connection with the database.

Action: Use ConnectOCL to connect to the database.

286000: Cannot disconnect while changes are pending for RDCI/RDCM work.

Severity: ERR

Cause: There is work in progress. It must either be erased or completed.

Action: Release the document using FlushRdciRdcm or WriteRdciRdcm depending upon whether or not you want to preserve the changes.

286100: RDCM with this received_dcm_id does not exist.

Severity: ERR

Cause: The RDCM cannot be located within the database.

Action: Query the stable interface or use GetRdcmArr to review all possible RDCMs.

286200: Status of Received DCM with this received_dcm_id is incompatible with data entry mode.

Severity: ERR

Cause: Incompatible modes of data entry - trying to update on a received document.

Action: Perform all required processes on the document before you begin to work in this mode.

286300: Locking mode incompatible with data entry mode.

Severity: ERR

Cause: Trying to update an unlocked document.

Action: Lock the document using FetchRdci in lock mode.

286500: Study is not enabled for data entry.

Severity: ERR

Cause: The study is not enabled to accept data.

Action: Enable the study for data entry in the design subsystem.

286600: Audit reason required and not provided.

Severity: ERR

Cause: A change was made to an accessible document but no audit reason was provided. The change may have been to a RDCI or RDCM key or comment, or to a response data, or to an investigator comment or by deleting a repeat.

Action: Return to the update and set a reason for the change using SetResponseData.

286700: Cannot locate the corresponding DCM question in the internal OCL buffers for the DCM question ID passed.

Severity: ERR

Cause: The question being asked does not belong to the selected DCM.

Action: Query the stable interface. See the *Oracle Clinical Stable Interface Technical Reference Manual* for the documented data model.

286800: No response in the internal buffers for the question.

Severity: ERR

Cause: Have not entered a response to the question.

Action: Use SetResponseData to enter a response.

286900: No existing univariate discrepancy for the question.

Severity: ERR

Cause: The data follows all business rules and does not include a discrepancy description.

Action: Use SetResponseData to enter a response.

287000: DCM question and DCM question group combination does not exist in DCM.

Severity: ERR

Cause: The question does not belong to the selected DCM.

Action: Query the stable interface. The *Oracle Clinical Stable Interface Technical Reference Manual* includes synonyms and views to access the data model.

287100: DCM question group does not exist in DCM.

Severity: ERR

Cause: The question does not belong to the selected DCM.

Action: Query the stable interface. The *Oracle Clinical Stable Interface Technical Reference Manual* includes synonyms and views to access the data model.

287200: Cannot insert repeats in browse mode.

Severity: ERR

Cause: Oracle Clinical is in a non-repeatable mode.

Action: Change the mode to a state that allows you to update.

287300: Invalid DCM question group ID passed.

Severity: ERR

Cause: DCM question group ID does not exist in the DCM.

Action: Query the stable interface for the correct ID or use the API GetQuestionId. The *Oracle Clinical Stable Interface Technical Reference Manual* includes synonyms and views to access the data model.

287400: Cannot insert repeat for a non-repeating question group.

Severity: ERR

Cause: This group is not set up to accept repeats.

Action: Modify the DCM definition in the Data Collection definition subsystem.

287500: Inserting repeat will introduce a null repeat in the middle.

Severity: ERR

Cause: A question has been skipping in a repeating question group.

Action: Do not skip repeats.

287600: Cannot insert for repeating defaults when repeat is less than the maximum repeats for DCM question group.

Severity: ERR

Cause: Attempt to insert repeats in a protected area.

Action: Ask someone who has the appropriate privileges to insert the repeat.

287700: The repeat exceeds the maximum repeats expected for DCM question group.

Severity: ERR

Cause: Number of repeats that have been input exceeds the number that the system was set to receive.

Action: If you have the appropriate privileges, modify the system to accept the additional repeats.

287800: Cannot delete repeats in browse mode.

Severity: ERR

Cause: Oracle Clinical is in a mode where you may not delete repeats.

Action: Change the mode to a state where you can modify the number of repeats.

287900: Cannot delete repeat for a non-repeating question group.

Severity: ERR

Cause: This group is not set up to accept repeat deletions.

Action: Modify the DCM definition in the Data Collection definition subsystem.

288000: Repeat sn does not exist for DCM question group.

Severity: ERR

Cause: The target row has not been created.

Action: Create the row first using InsertRepeat.

288100: Cannot delete for repeating defaults when repeat is less than the maximum repeats for DCM question group.

Severity: ERR

Cause: Attempt to insert repeats in a protected area.

Action: Ask someone who has the appropriate privileges to insert the repeat.

288300: The API RDCI record is not locked.

Severity: ERR

Cause: The system cannot update the record because it is not locked.

Action: Lock the header and then update the record.

288400: The completion flag is not compatible with the data entry mode.

Severity: ERR

Cause: You are in a mode where you are not allowed to save a partially completed DCM.

Action: Flush all changes and return to finish the work when you have all the required data.

288500: Write responses called but no changes pending.

Severity: WRN

Cause: The user has indicated that he or she wants to commit but the system has received no data.

Action: Warning message; no action necessary.

288600: Blank repeats exist.

Severity: ERR

Cause: The repeats to do not contain any information.

Action: Either delete the repeat using DeleteRepeat or add data using SetResponseData.

288700: Mandatory field missing.

Severity: ERR

Cause: There is missing data.

Action: Provide data using SetResponseData or, if the Data Entry mode permits, do an incomplete comment with WriteResponses.

An example of an incomplete comment: "Do not know the answer. Will complete this question later."

288800: Cannot set actual event when RDCI is locked.

Severity: ERR

Cause: The document is locked and the parent record is in a state of change.

Action: Release the document using FlushRdciRdcm or WriteRdciRdcm depending upon whether you want to preserve the changes.

Fetch the record from the database using FetchRdci.

Finish modifying the parent record and put it in the database or release the lock and state that there is no change.

288900: Cannot enroll a patient when RDCI is locked.

Severity: ERR

Cause: The document is locked and the parent record is in a state of change.

Action: Release the document using FlushRdciRdcm or WriteRdciRdcm depending upon whether you want to preserve the changes.

Fetch the record from the database using FetchRdci.

Finish modifying the parent record and put it in the database or release the lock and state that there is no change.

289000: Cannot set page status when RDCI is locked.

Severity: ERR

Cause: The document is locked and the parent record is in a state of change.

Action: Release the document using FlushRdciRdcm or WriteRdciRdcm depending upon whether you want to preserve the changes.

Fetch the record from the database using FetchRdci.

Finish modifying the parent record and put it in the database or release the lock and state that there is no change.

289100: Unable to find the RDCI to which page belongs: \0.

Severity: ERR

Cause: Invalid received page ID passed to SetPageStatus.

Action: Pass correct received_page_ID.

289200: Unable to lock the RDCI to which page belongs.

Severity: ERR

Cause: Someone else is using the RDCI.

Action: Wait a few minutes and attempt to lock the RDCI again.

289300: 'Unknown' page status can only be changed to 'Missing' or 'Present'.

Severity: ERR

Cause: Invalid page status passed to SetPageStatus.

Action: Use other means to update the status.

289400: The page status cannot be manually changed to a system status.

Severity: ERR

Cause: Invalid page status passed to SetPageStatus.

Action: Use other means to update the status.

289500: A system page status cannot be manually changed.

Severity: ERR

Cause: Invalid page status passed to SetPageStatus.

Action: Use the correct combination which is provided by the error message.

289800: This document already exists in the database.

Severity: ERR

Cause: Attempting to create a document with the same set of keys as an existing document.

Action: Change keys to create a new document or update an existing document.

289900: Document number not derived.

Severity: ERR

Cause: User has not provided the document number. The system cannot derive a document number because the user has not set the scheme to devise a document number in OCL.

Action: Either provide the document number or provide a scheme to devise one.

289000: Duplicate Received DCI found with the same keys in document number: \0.

Severity: ERR

Cause: The document that is about to be written to the database has been created by another user in the meantime.

Action: Roll back and start again.

289100: Duplicate Received DCM found within this Received DCI.

Severity: ERR

Cause: The document that is about to be written to the database has been created by another user in the meantime.

Action: Roll back and start again.

289200: Duplicate Received DCM found in document number: \0.

Severity: ERR

Cause: The document that is about to be written to the database has been created by another user in the meantime.

Action: Roll back and start again.

289300: Current record is not complete: Operation was not successful.

Severity: ERR

Cause: Called ProcessRdci or ProcessRdcm before providing all required keys.

Action: Provide the missing keys in order to proceed.

289500: This site/investigator pair is invalid for study.

Severity: ERR

Cause: The investigator is not assigned to this site.

Action: Either change the investigator assignment or the investigator.

289600: Invalid investigator.

Severity: ERR

Cause: The investigator is not assigned to this study.

Action: Either create the investigator for the study or correct the data in the document.

289700: Invalid site.

Severity: ERR

Cause: The site is not assigned to this study.

Action: Either create the site for the study or correct the data in the document.

289800: Patient position is frozen.

Severity: ERR

Cause: Attempt to add data for a patient that is frozen. The site is not assigned to this study.

Action: Unfreeze the patient before adding data.

289900: Patient not enrolled for study.

Severity: WRN

Cause: The patient has not been enrolled in the study.

Action: Enroll the patient in the study using EnrollPatient.

291000: Patient position is invalid.

Severity: ERR

Cause: The patient has not been created in the study.

Action: Add the patient to the study using the Oracle Clinical design subsystem or correct the data in the document.

291100: Patient is not associated with a valid site (active site/site role).

Severity: WRN

Cause: The patient has not been enrolled in the study.

Action: Assign the patient to a valid site using the Oracle Clinical design subsystem or correct the data in the document.

291200: DCI is inactive or invalid.

Severity: ERR

Cause: Attempt to add data for invalid or inactive patient.

Action: Select a different DCI or, if appropriate, activate the DCI.

291300: [error message returned by OCL_CLIENT_PACK. Validate document]

Severity: ERR

Cause: This internal error can display different error messages.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

291400: Clinical planned event is invalid for the current study.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

291500: Subevent not found: actual event will be created once you save.

Severity: WRN

Cause: This is an informational message.

Action: You may wish to set the actual event using SetActualEvent.

291600: Invalid lab.

Severity: ERR

Cause: This is an informational message.

Action: You may wish to add the labs using the labs menu in Oracle Clinical.

291700: Discrete value does not exist or is inactive.

Severity: ERR

Cause: The discrete value used for validation is invalid.

Action: Create or activate a discrete value.

291800: Discrete value could not be validated.

Severity: ERR

Cause: The discrete value used for validation is invalid.

Action: Run a query in the Maintain Discrete Value Groups window in Oracle Clinical (Glib, Discrete Value Grps, Discrete Value Grps) to see if this is the correct discrete value group.

291900: Value must be Y or N.

Severity: ERR

Cause: Incorrect value.

Action: Enter Y or N.

292000: Blank flag must be entered.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292100: Clinical planned event name must be entered.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292200: Patient must be entered.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292300: DCI name must be entered.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292400: Site is required.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292500: Clinical planned event name is required.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292600: Patient is required.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292700: DCI is required.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292800: DCI date must be entered.

Severity: ERR

Cause: This is a mandatory field that may not be left blank or null.

Action: Input the appropriate value.

292900: Error soft deleting actual event.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

293000: Subevent number must be entered.

Severity: ERR

Cause: This is a mandatory flag that may not be left blank or null.

Action: Input the appropriate value.

293300: DCM time must be entered.

Severity: ERR

Cause: This is a mandatory flag that may not be left blank or null.

Action: Provide requested information.

293400: Qualifying value must be entered.

Severity: ERR

Cause: This is a mandatory flag that may not be left blank or null.

Action: Provide requested information.

293500: Cannot set to N (not blank) when Received DCI is blank.

Severity: ERR

Cause: Attempt to enter data for a Received DCM whose parent RDCI is blank.

Action: Either correct the parent Received DCI or choose another Received DCM.

293600: Too many actual events correspond to this subevent #.

Severity: ERR

Cause: Internal error.

Action: Contact Oracle Clinical support.

293700: DCM date must be entered.

Severity: ERR

Cause: A mandatory field is blank.

Action: Provide DCM date in order to proceed.

293800: You are working with a data locked record and have privileged update.

Severity: WRN

Cause: When a record is data locked, the information cannot be changed or deleted by anyone who does not have privileged update. This is an informational message to remind you that you have the privilege to update the data locked record.

Action: N/A.

294000: An error occurred while soft-deleting the responses.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

294200: An error occurred while processing the Received DCI structure and updating the PATIENT_DM_TRACKING table.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

294300: An error occurred while processing the Received DCM structure and updating the received DCI.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

294500: Document number already exists with patient from another location.

Severity: ERR

Cause: This is a duplicate.

Action: Change the patient or correct the location.

294600: Document found in different study. Change study to access.

Severity: ERR

Cause: The target document exists in another study.

Action: Point to the appropriate study to continue.

294700: Received DCI is protected because at least one of its received DCMs is locked.

Severity: WRN

Cause: Cannot access DCI because a DCM is locked.

Action: The DCI must be updated by an approved super user who has the appropriate privileges.

294800: An error occurred while in the Rxc_Login_Pack *SoftDeleteAll* received DCMs function.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

294900: Site is currently inactive.

Severity: WRN

Cause: Site is currently inactive.

Action: Activate the site.

295000: Investigator is currently inactive.

Severity: WRN

Cause: Investigator is currently inactive.

Action: Activate the investigator.

295100: Site/investigator pair is not current in study.

Severity: WRN

Cause: Site/investigator pair is not current in study.

Action: Create an assignment for this site/investigator pair.

295200: Too many rows found in NON_EXPECTED_DCMS table.

Severity: ERR

Cause: Internal error.

Action: Contact Oracle Clinical support.

295300: An error occurred querying DVGs for an RDCM.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

295400: An error occurred validating the clinical planned event.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

295800: Problem retrieving from the actual event sequence.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

296100: Too many DCI IDs for DCI short name.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

296200: Too many DVGs match DVG ID.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

296300: Too many labs with same name.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

296600: Cannot preserve lock when performing a rollback.

Severity: ERR

Cause: Cannot preserve lock when performing a rollback.

Action: Either commit or do not preserve the lock.

296700: Unable to lock RDCI.

Severity: ERR

Cause: Another user is working on the RDCI.

Action: Come back later and retry.

297000: Null or invalid input parameters provided.

Severity: ERR

Cause: Null or invalid input parameters provided.

Action: Provide valid inputs with proper memory allocation to APIs.

297100: Changes pending.

Severity: ERR

Cause: Cannot take the actions because there are changes that are in process.

Action: Commit changes to the database.

297200: Invalid execution context.

Severity: ERR

Cause: The execution context provided was not one of the following: SINGLE, ON-LINE, ON-LINE/DCM.

Action: Use one of the following valid execution context values: SINGLE, ON-LINE, ON-LINE/DCM. Enter them exactly as they are listed above.

297300: Procedure ID is required for SINGLE execution context.

Severity: ERR

Cause: No Procedure ID was provided with the SINGLE execution context.

Action: Provide a Procedure ID.

297400: Procedure version is required for SINGLE execution context.

Severity: ERR

Cause: No Procedure Version was provided with the Single execution context.

Action: Provide a Procedure version.

297500: Invalid Procedure (nonexistent, retired or pre-V3.1-style).

Severity: ERR

Cause: The provided Procedure does not exist, is retired, or is not V3.1-style.

Action: Provide a valid Procedure that exists, is not retired, and is in V3.1-style.

297600: Procedure is not active or needs to be regenerated.

Severity: WRN

Cause: The Procedure provided for OCL_PROD execution was not active or needs regeneration.

Action: Provide an active Procedure that does not need to be regenerated.

297700: Procedure is not compiled.

Severity: WRN

Cause: The Procedure provided for OCL_TEST execution is not compiled.

Action: Provide an active Procedure that does not need to be regenerated.

297800: DCM ID is required for ON-LINE/DCM execution context.

Severity: ERR

Cause: No DCM ID was provided with the ON-LINE/DCM execution context.

Action: Provide a DCM ID.

297900: DCM ID does not apply for ON-LINE execution context.

Severity: ERR

Cause: A DCM ID was provided with the ON-LINE execution context. It is only applicable for the ON-LINE/DCM execution context.

Action: Provide -1 as the DCM ID value.

298000: DCM ID does not apply for SINGLE execution context.

Severity: ERR

Cause: A DCM ID was provided with the SINGLE execution context. It is only applicable for the ON-LINE/DCM execution context.

Action: Provide -1 as the DCM ID value.

298100: Procedure ID and version must be -1 for ON-LINE and ON-LINE/DCM execution contexts.

Severity: ERR

Cause: Procedure ID and version do not apply for ON-LINE or ON-LINE/DCM execution context.

Action: Provide -1 as the Procedure ID and version values.

298200: Mandatory patient position ID is missing.

Severity: ERR

Cause: Patient position ID is a required argument.

Action: Provide a Patient position ID.

298300: Invalid preserve lock value.

Severity: ERR

Cause: Preserve Lock value is neither TRUE (1) nor FALSE (0).

Action: Provide TRUE (1) or FALSE (0) as the Preserve Lock value.

298400: Mandatory clinical study ID is missing.

Severity: ERR

Cause: Clinical Study ID is a required argument.

Action: Provide a clinical study ID.

298500: Mandatory clinical study version ID is missing.

Severity: ERR

Cause: Clinical Study Version ID is a required argument.

Action: Provide a clinical study Version ID.

298600: Invalid mode value.

Severity: ERR

Cause: Mode is neither OCL_PROD (0) nor OCL_TEST (1), 2, or 3.

Action: Provide OCL_PROD (0) or OCL_TEST (1), 2, or 3 as the mode value.

Note: A mode value of 2 is used for Production and 3 for Test mode when invoking the ConnectOCL DCAPI function to share the same database session.

298700: Invalid debug flag value.

Severity: ERR

Cause: Debug flag value is neither TRUE (1) nor FALSE (0).

Action: Provide TRUE (1) or FALSE (0) as Debug flag value.

298800: Mandatory execution context is missing.

Severity: ERR

Cause: Execution context is a required argument.

Action: Provide an execution context.

299000: Only browse mode can be performed on a frozen study.

Severity: ERR

Cause: Attempt to access study in mode other than Browse.

Action: Correct mode, choose different study or unfreeze study.

299100: No data has been entered for this RDCM.

Severity: ERR

Cause: You are attempting to initialize RDCM responses but no data has been entered for the RDCM.

Action: Enter data for the RDCM.

299200: RDCI status is incompatible with querying the record in first-pass entry mode.

Severity: ERR

Cause: First pass entry has already been performed for this entry.

Action: Choose a different document or switch to update mode.

299300: RDCI is accessible. You can query only inaccessible documents in first-pass entry mode.

Severity: ERR

Cause: You have selected an accessible document.

Action: Choose a different document or switch to update mode.

299400: Operator comment in browse is disabled. You cannot fetch the document in locking mode.

Severity: ERR

Cause: Operator comment is disabled.

Action: Fetch the document in non-locking mode or enable operator comment in Browse mode.

299500: RDCI is inaccessible. You can query only accessible documents in browse mode.

Severity: ERR

Cause: The RDCI you selected is inaccessible, so you cannot query for it in Browse mode.

Action: Choose a different document or switch to First-pass Data Entry mode.

299600: Owning location of the patient is different from that of the database.

Severity: ERR

Cause: Invalid patient.

Action: Use a different patient or enter data for this patient in a different database.

299700: The data entered for this received DCI will be deleted.

Severity: WRN

Cause: Informational message.

Action: N/A.

299800: Date cannot exceed today's date.

Severity: ERR

Cause: Attempt to enter a document at a future date.

Action: Correct the date.

299900: Cannot proceed when blank flag is set to Y.

Severity: ERR

Cause: Attempt to enter data into a blank document.

Action: Either set blank to N or skip the document.

300100: Only the user who performed Pass 1 Entry can perform data entry task.

Severity: ERR

Cause: Although Pass 1 Entry has already been performed, another user is attempting the task.

Action: Have the original user update the document.

300200: This patient is frozen.

Severity: ERR

Cause: Attempt to enter data for a frozen patient.

Action: Have a user with the appropriate privileges unfreeze the patient.

300300: This Received DCI is data locked.

Severity: ERR

Cause: Attempt to update a locked RDCI.

Action: Have a user with the appropriate privileges perform the update or unlock the document.

300500: First pass entry cannot be performed on an accessible RDCM.

Severity: ERR

Cause: Attempt to perform First Pass Entry on an accessible RDCM.

Action: Choose a different document or switch to update mode.

300600: RDCI status not compatible with the mode.

Severity: ERR

Cause: There are many instances when this error may occur, depending upon the status of the document and the mode of data entry. As an example, First Pass Entry has already been performed for the selected document.

Action: Choose a different document (which has a different status) or switch to a different mode. In this example, you can either choose a document with a status of *Received* or *Pass 1 Started*. You may then perform first pass on it, or you can update the document because it has already had first pass performed on it.

300700: Patient is frozen. The RDCI record will behave as if it were fetched in non-locking mode.

Severity: WRN

Cause: Informational message.

Action: Unfreeze the patient in order to proceed.

300800: Data on this RDCI is locked. The RDCI record will behave as if it were fetched in non-locking mode.

Severity: WRN

Cause: Informational message.

Action: No action necessary.

300900: No update allowed to the RDCI buffer.

Severity: ERR

Cause: The RDCI is not locked so you cannot update the RDCI buffer.

Action: Fetch the RDCI with a lock if you have the authority.

301200: No changes to save.

Severity: WRN

Cause: Informational message.

Action: No action necessary.

301300: Invalid clinical study.

Severity: ERR

Cause: Attempt to access a study that is invalid or inactive.

Action: Choose a different study or create a live version of the study.

301400: The clinical study is not live.

Severity: ERR

Cause: Attempt to access a study that is invalid or inactive.

Action: Choose a different study or create a live version of the study.

301500: Data entry is not enabled for the clinical study.

Severity: ERR

Cause: Attempt to enter data for a study that is not enabled.

Action: Enable data entry in order to enter data.

301600: Clinical study does not exist in the current location.

Severity: ERR

Cause: Attempt to choose a study that is from a different location.

Action: Go to the correct location to enter data.

301700: No study state for the clinical study.

Severity: ERR

Cause: The clinical study does not have a study state.

Action: Create a study state.

301800: No study access account exists for the study.

Severity: ERR

Cause: The clinical study does not have a study access account.

Action: Create a study access account.

301900: Record validation failed.

Severity: ERR

Cause: Validation of the study record failed so you can not set context to that study.

Action: Try another study or see your system administrator.

302000: User is not authorized to access the study.

Severity: ERR

Cause: The user does not have the appropriate privileges.

Action: Either grant privileges to this user or have a user with the appropriate privileges perform the work.

302100: Validation of study failure.

Severity: ERR

Cause: Invalid entry.

Action: Provide a valid study.

302200: Mode should be OCL_PROD or OCL_TEST.

Severity: ERR

Cause: Invalid mode provided.

Action: Provide correct mode.

302300: Cannot disconnect while changes are pending for responses work.

Severity: ERR

Cause: Changes are in progress.

Action: Either save or flush and then disconnect.

302400: Zero rows updated.

Severity: WRN

Cause: Informational message.

Action: No action necessary.

302500: Invalid page status.

Severity: ERR

Cause: Invalid page status passed to SetPageStatus.

Action: Use other means to update the status.

302600: Cannot delete when repeat sn is greater than the number of repeats currently existing for the question group.

Severity: ERR

Cause: The repeat that was the deletion target does not exist.

Action: N/A.

302700: No existing manual discrepancy for the question.

Severity: ERR

Cause: Attempt to access a manual discrepancy for a response that does not have a manual discrepancy.

Action: Choose another response with a manual discrepancy or create one for this response.

302800: Invalid discrepancy review status code.

Severity: ERR

Cause: An incorrect discrepancy code was entered.

Action: Look up the appropriate code in the reference codelist.

302900: Invalid discrepancy resolution type code.

Severity: ERR

Cause: An incorrect discrepancy resolution type code was entered.

Action: Look up the appropriate code in the reference codelist.

303000: Resolution type code cannot be null for RESOLVED review status.

Severity: ERR

Cause: The resolution type code has not been input.

Action: Look up the appropriate code in the reference codelist.

303100: Resolution type code should be null if review status is not RESOLVED.

Severity: ERR

Cause: A resolution type code has been provided. The resolution type code should be NULL.

Action: Do not provide the resolution type code.

303200: Invalid manual discrepancy type.

Severity: ERR

Cause: An invalid manual discrepancy type was entered.

Action: Look up the appropriate code in the reference codelist.

303300: Changes are pending for RDCI/RDCM work.

Severity: ERR

Cause: Change of context will result in loss of data.

Action: Commit or rollback explicitly.

303400: Changes are pending for responses work.

Severity: ERR

Cause: Change of context will result in loss of data.

Action: Commit or rollback explicitly.

303500: Document contains empty repeats.

Severity: ERR

Cause: Data was not provided for all repeats before writing.

Action: Enter data for all repeats before writing.

303600: Invalid patient.

Severity: ERR

Cause: An invalid patient was entered.

Action: Provide a valid patient ID.

303700: Patient does not belong to the current location.

Severity: ERR

Cause: Patient does not belong to the current location.

Action: Either correct the location or do not enter dates for this part.

303800: Cannot find location_code in the reference codelist for the current connection.

Severity: ERR

Cause: The reference codelist is incorrect.

Action: Correct the reference codelist.

303900: Too many current documents with this number.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304000: Too many patients with this name.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304100: Too many occurrences of this CPE name exist.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304200: Too many sites with this name.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304300: Too many investigators.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304400: No procedures qualify for execution.

Severity: WRN

Cause: No target Procedures were found to match the specified execution criteria, due to one or more of the following causes:

- target Procedure(s) are retired, not compiled, not V3.1-style, or are in need of regeneration.

- if requested mode was OCL_PROD, target Procedure(s) are not active.
- if requested execution context was ON-LINE, no Procedure(s) classified as ON-LINE or ON-LINE/DCM were found.
- if requested execution context was ON-LINE/DCM, no Procedure(s) classified as ON-LINE/DCM with Primary Reference Question Group DCM ID matching the input argument DCM ID were found.
- if requested execution context was SINGLE, no match on input arguments Procedure ID/Version was found.

Action: Target Procedure(s) are identified for execution according to the following requirements. Procedure(s) must:

- be compiled.
- not be retired or in need of regeneration.
- be V3.1-style.
- if requested mode is OCL_PROD, Procedure(s) must be active.
- if requested execution context is ON-LINE, Procedure(s) must be classified as ON-LINE or ON-LINE/DCM.
- if requested execution context is ON-LINE/DCM, Procedure(s) must be classified as ON-LINE/DCM with a Primary Reference Question Group DCM ID that matches the input argument DCM ID.
- if requested execution context is SINGLE, a match on Procedure ID/Version must exist.

304500: Cannot preserve lock as the Received DCI is not locked.

Severity: ERR

Cause: The received DCI is not locked.

Action: Do not try to preserve the lock. If you want to lock the document, call the appropriate function.

304600: Function does not exist.

Severity: ERR

Cause: In attempting to establish whether you have the privilege for a specific function, it has been determined that the supplied function name is invalid. This may be because important OCL initializations have not taken place.

Action: See your system administrator.

304700: Privilege does not exist.

Severity: ERR

Cause: Internal error.

Action: Print a copy of the entire error stack and contact Oracle Clinical support. Be prepared to describe your most recent actions and to provide sample code.

304800: User is not authorized to call this function.

Severity: ERR

Cause: The user does not have the privilege to call this function.

Action: Grant the user the appropriate privilege to be able to call this function.

304900: User has been granted too many roles: exceeds the maximum allowed.

Severity: WRN

Cause: Informational message.

Action: Reduce the number of roles for the user or choose another user for the role.

305000: Too many messages.

Severity: WRN

Cause: The system has not retrieved all messages because the error stack is full.

Action: None. This is just an informational message.

305600: Invalid DCI date.

Severity: ERR

Cause: Date is invalid.

Action: Input the correct date in the proper format.

305700: Invalid DCI time.

Severity: ERR

Cause: Time is invalid.

Action: Input the correct time in the proper format.

305800: Invalid DCM date.

Severity: ERR

Cause: Date entered is invalid.

Action: Input the correct date in the proper format.

305900: Invalid DCM time.

Severity: ERR

Cause: Time entered is invalid.

Action: Input the correct time in the proper format.

306100: Changes pending that have not been processed.

Severity: ERR

Cause: You have not yet processed outstanding changes.

Action: Call ProcessRdci to process outstanding changes.

306200: RDCI record being deleted does not exist in the database.

Severity: ERR

Cause:

Action: Choose another record to delete.

306300: Received DCI is inactive or invalid.

Severity: ERR

Cause: You attempted to activate a received DCI that is inactive or invalid.

Action: Choose another record to activate.

306400: Too many current actual events.

Severity: ERR

Cause: Internal error.

Action: Contact Oracle Clinical support.

306500: Unable to release lock.

Severity: ERR

Cause: Internal error.

Action: Contact Oracle Clinical support.

306600: This received DCM is locked.

Severity: ERR

Cause: The data for the RDCM is locked and you are:

- trying to initialize RDCM responses but you are in Update without appropriate privileges
- not in Update or Browse mode

Action: Either switch to Browse mode or have your administrator give you Update privilege.

306700: Cannot set RDCI time when name is not set.

Severity: ERR

Cause: Attempt to set RDCI time without providing the name of the visit.

Action: Provide the name of visit before attempting to set RDCI time.

306800: Invalid audit reason.

Severity: ERR

Cause: The user-entered audit reason is not present in the reference codelist.

Action: Refer to the reference codelist for a list of valid audit reasons.

306900: No update allowed to the RDCM buffer.

Severity: ERR

Cause: The parent RDCI of the RDCM is not locked so you can not update the RDCM buffer.

Action: If you have the authority, fetch the RDCI with a lock.

307000: Cannot call this function if you have not locked the record.

Severity: ERR

Cause: Attempt to call function without locking the record.

Action: Lock the RDCI before calling this function.

307100: Investigator is required.

Severity: ERR

Cause: The name of the investigator is a mandatory value that must be entered in order to proceed.

Action: Provide the name of the investigator.

307200: DCI Time is required.

Severity: ERR

Cause: The DCI Time must be entered in order to proceed.

Action: Provide the DCI Time.

307400: Please use the patient enrollment form to specify which DCI Book to use to maintain the page tracking information.

Severity: WRN

Cause: No DCI Book has been assigned to the patient.

Action: Assign a DCI Book for the patient.

307600: Can not log-in data for a patient that is not associated with a book in a study with no default DCI book.

Severity: ERR

Cause: Although page tracking is enabled for the DCI and the study, both the patient and the study do not have an assigned DCI book.

Action: Assign a DCI book at the study level or assign a DCI book to the patient.

308200: The DCI book assigned to this patient is not active. Can not use a DCI book that is not active to log data in a production environment.

Severity: ERR

Cause: Although a DCI book has been assigned, it has not been activated.

Action: Activate the DCI book or assign an activated DCI book to the patient.

308500: The current DCI book is Retired and the environment is Test.

Severity: ERR

Cause: This combination is not allowed.

Action: Either use active or run manual DCI books.

308600: Record changed by another user.

Severity: ERR

Cause: Another user has changed the record since you fetched it.

Action: Fetch the record again.

308700: Cannot set RDCI time until RDCI short name is set.

Severity: ERR

Cause: You attempted to set the RDCI time before setting the RDCI short name.

Action: First provide DCI name, then set the time.

308800: DCI short name is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

308900: DCI short name is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

309000: Document number is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

309100: Document number is not updateable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

309200: Patient is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

309300: Patient is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

309400: Date is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

309500: Date is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

309600: Time is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

309700: Time is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

309800: Visit name is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

309900: Visit name is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

310000: Subevent number is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

310100: Subevent number is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

310200: Site is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

310300: Site is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

310400: Investigator is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

310500: Investigator is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

310600: Blank flag is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

310700: Blank flag is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

310800: Comment is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

310900: Comment is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

311000: Qualifying value is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

311100: Qualifying value is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

311200: Lab is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

311300: Lab is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

311400: Data comment is not applicable.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

311500: Data comment is not updateable.

Severity: ERR

Cause: You can not update an existing value for an existing record for this field.

Action: Do not attempt to provide this field.

311600: DCI document number can not be updated to null.

Severity: ERR

Cause: You can not provide a value for this field in a new record.

Action: Do not attempt to provide this field.

311700: Document number must not have any lower case letters.

Severity: ERR

Cause: The document number includes lower-case letters, which Oracle Clinical prohibits in document numbers.

Action: Change the document number so that all its letters are upper-case.

311800: Patient not assigned to this site.

Severity: WRN

Cause: The patient is not assigned to this site.

Action: No action necessary. Verify that you have selected the correct site.

312200: Can not delete Received DCI because it is data locked.

Severity: ERR

Cause: Attempt to delete a locked RDCI record.

Action: Because the data on this record is locked, it cannot be changed or deleted. A system administrator must unlock this record before it can be deleted.

312500: Can not update repeating defaults when repeat sn is less than the maximum repeats expected for DCM question group.

Severity: ERR

Cause: Attempt to update the protected repeating default. You can not update because you do not have privileged update.

Action: The data must be updated by someone who has privileged update.

313600: You have privileged update and have deleted repeating default when repeat sn is less than the maximum repeats for DCM question group.

Severity: WRN

Cause: Superuser has attempted to modify a protected repeating field.

Action: Restore deleted field, if desired.

313800: User is not authorized to call ExecuteMultivariate (not granted RXCLIN_MOD privilege)

Severity: ERR

Cause: The user requires the RXCLIN_MOD privilege to call the ExecuteMultivariate function.

Action: Grant this privilege to this user.

314800: Received DCI is accessible; please use Key Changes to change Blank Flag

Severity: ERR

Cause: You are attempting to change the Blank Flag for an accessible document.

Action: Change mode to KEY CHANGES to change the Blank Flag.

314900: Received DCM is accessible; please use Key Changes to change Blank Flag

Severity: ERR

Cause: You are attempting to change the Blank Flag for an accessible Received DCM.

Action: Change mode to KEY CHANGES to change the Blank Flag.

315000: Some received DCMs are accessible; please use Key Changes to change Blank Flag

Severity: ERR

Cause: You are attempting to change the Blank Flag for a document that has some accessible Received DCMs.

Action: Change mode to KEY CHANGES to change the Blank Flag.

315100: Responses exist for this RDCI. Changing the Blank Flag to 'Y' will cause them to be irrevocably deleted

Severity: WRN

Cause: The document has responses entered. If the RDCI Blank Flag is changed to 'Y' then all responses entered for this document will be deleted. This is an informational message to warn you of the consequence of your action.

Action: Do not change the RDCI Blank Flag to 'Y' if you want to retain the responses entered.

315200: Responses exist for this RDCM. Changing the Blank Flag to 'Y' will cause them to be irrevocably deleted

Severity: WRN

Cause: The Received DCM has responses entered. If the RDCM Blank Flag is changed to 'Y' then all responses entered for Received DCM will be deleted. This is an informational message to warn you of the consequence of your action.

Action: Do not change the RDCM Blank Flag to 'Y' if you want to retain the responses entered.

315500: RDCI is not protected because at least one of its RDCMs is locked and you have privileged update.

Severity: WRN

Cause: When a record is data locked, the information cannot be changed or deleted by anyone who does not have privileged update. This is an informational message to remind you that you have the privilege to update the data locked record.

Action: N/A

317400: Another session is connected. Cannot switch database modes.

Severity: ERR

Cause: Oracle Clinical prevents you from switching database roles when you have multiple sessions open.

Action: Close all other running sessions, then switch roles.

317800: Cannot delete accessible Received DCI in Initial Login Mode.

Severity: ERR

Cause: You are attempting to delete an accessible document by fetching the document in Initial Login mode.

Action: Fetch the document in Key Changes mode.

345400: Qualifying value can only be changed to the default value defined at the DCI module.

Severity: ERR

Cause: You are attempting to change the qualifying value to a value other than the default value.

Action: Set the qualifying value to the value specified as the default.

659700: No default audit reason.

Severity: ERR

Cause: No default audit reason was found in either of the reference codelists.

Action: Supply a default audit reason in one of the two reference codelists. See "Auditing and Backward Compatibility" on page 5-3 for an explanation of how the system uses reference codelists to supply default audit change reasons.

659800: More than one default audit reason found.

Severity: ERR

Cause: Both reference codelists had a default value.

Action: Only one of the two reference codelists should have a default audit reason. Remove the default audit reason from one of the codelists. See "Auditing and Backward Compatibility" on page 5-3 for an explanation of how the system uses reference codelists to supply default audit change reasons.

660000: Cannot use system generated audit reason.

Severity: ERR

Cause: User has provided a system-generated audit reason as the user-entered audit reason.

Action: Provide an audit reason that is valid in the reference codelists.

685900: Audit comment for Deletion of Repeat cannot have more than 158 bytes.

Severity: ERR

Cause: The audit comment provided for DeleteRepeat is longer than 158 characters.

Action: Provide an audit comment that is 158 characters or shorter.

743300: Cannot delete Received DCI without a valid audit reason.

Severity: ERR

Cause: DeleteRdci failed because the user-entered audit reason is not present in the reference codelist.

Action: Refer to the reference codelist for a list of valid audit reasons.

970100: The following sequence(s) is approaching itx maximum value. Reset each listed sequence to an unused lower value.

Severity: ERR

Cause: When Oracle Clinical processes modules where new received DCIs, received DCMs, discrepancies and responses are created—Data Entry, Discrepancy Management, Mass Change, Batch Validation, Procedures—it checks that these sequences are not close to their maximum limits. When the system identifies a sequence that will exceed its limits, risking data corruption, this error message is issued.

Action: Reseed the sequence or sequences listed in the error message. Refer to the *Oracle Clinical Administrator's Guide* for information on how to do this.

992100: All other parameters except the resolution text in the discrepancy structure will be ignored.

Severity: WRN

Cause: A parameter other than resolution text contained a value.

Action: Ensure that all parameters other than resolution text are initialized but contain only the empty string.

992300: Unable to get existing discrepancy information from the database.

Severity: ERR

Cause: The function only acts on current discrepancies.

Action: Make sure the discrepancy has a status of CURRENT.