Table of Contents

[Screen Shot of EXTRACT2 2](#_Toc324159460)

[Forms Startup 2](#_Toc324159461)

[Main Menu (#1) 2](#_Toc324159462)

[Exit Button (#2) 3](#_Toc324159463)

[Selection Criteria (#3) 3](#_Toc324159464)

[Action Buttons/Extraction Type (#4) 3](#_Toc324159465)

[Create Extract Button 3](#_Toc324159466)

[Generate Text Files Button 5](#_Toc324159467)

[Transfer to Archive Table Button 5](#_Toc324159468)

[Remove Last Extract Button 6](#_Toc324159469)

[Status (#5) 6](#_Toc324159470)

[Extract Status (#6) 6](#_Toc324159471)

[Error Status (#7) 6](#_Toc324159472)

[Refresh Button (#8) 7](#_Toc324159473)

[Extract Status Detail (#9) 7](#_Toc324159474)

[Error Status Detail (#10) 7](#_Toc324159475)

[Security Detail (#10) 7](#_Toc324159476)

[TRIGGER CODE 8](#_Toc324159477)

[CREATE EXTRACT 8](#_Toc324159478)

[TRANSFER TO ARCHIVE TABLE 11](#_Toc324159479)

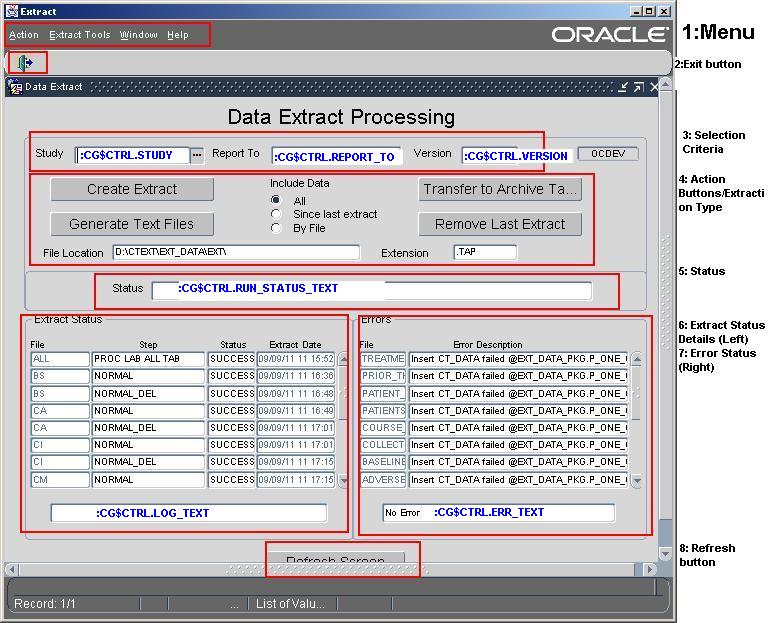
[GENERATE TEXT FILES 12](#_Toc324159480)

[REMOVE LAST EXTRACT 17](#_Toc324159481)

Detailed analysis of EXTRACT02.FMB

Note: Used TEXTPAD to look inside form as the 6i Form Builder was unable to read form.

# Screen Shot of EXTRACT2



# Forms Startup

1. Security Checked at form startup. The following query checks userid to see if user is valid for APPLICATION.

SELECT '1' INTO T\_WHAT

FROM CT\_EXT\_ACCOUNTS

WHERE (USER\_NAME = USER or User = 'CTEXT\_T')

AND ROWNUM = 1;

1. Action Buttons are turned OFF if the user already has a JOB running in ORACLE where the job running is name like “EXT\_DATA\_PKG.P\_EXTRACT%”

# Main Menu (#1)

The main menu controls what other screens can be selected. They are controlled through ROLES. Only certain Oracle ROLES granted to the user would allow access. See EXTMENU1\_Analysis for more information.

# Exit Button (#2)

The exit button is used to close the application. When closing the application, the screen is checked for existing non-committed transactions and will not quite until the transactions are committed or rolled back.

# Selection Criteria (#3)

There are 3 fields that make up the selection criteria, Study, Report To and Version. They each have a drop down that is based of the previous item

1. STUDY: The list of available studies is based on the following query:

SELECT distinct OC\_STUDY FROM CT\_EXT\_STUDY\_SEC\_VW ORDER BY 1

1. REPORT TO: This is list of report to values based upon entered study:

SELECT distinct REPORT\_TO, VERSION FROM CT\_EXT\_STUDY\_RPT\_CTL WHERE OC\_STUDY = :CG$CTRL.STUDY ORDER BY 1

NOTE: This query actually does both REPORT TO and VERSION, This must be broken up.

1. VERSION: This is the version of the REPORT TO of the STUDY. Use this query:

SELECT distinct VERSION FROM CT\_EXT\_STUDY\_RPT\_CTL WHERE OC\_STUDY = :CG$CTRL.STUDY and REPORT TO = :CG$CTRL.REPORT\_TO ORDER BY

# Action Buttons/Extraction Type (#4)

#### Create Extract Button

The Create Extract Button is used to create a data set of extract study data from C3D. It is the primary function of this application.

1. Only fires if Study/Report To/Version are entered.
2. Checks Archive Data Table to see if extract has been performed already. User must acknowledge to append data to existing data. Use the following queries to identify this problem:

-- Gets latest Extract date from archive

SELECT MAX(EXTRACTED) FROM CT\_DATA WHERE PROTOCOL = :CG$CTRL.STUDY;

-- Gets Extract date from “To be Extracted” data

SELECT MAX(CURRENT\_EXT\_DATE) FROM CT\_EXT\_FILE\_CTL WHERE OC\_STUDY = :CG$CTRL.STUDY;

If dates are equal then there is a problem and the user must FORCE data to be appended or they can simple REMOVE the previous data with the REMOVE LAST EXTRACT Button .

1. For CDUS Extraction, user must select COMPLETE or ABBREVIATED Extraction.
2. File Location and Extension are values held in the table CT\_APP\_META\_DATA where the name equals “FILEDIR” and “FILEEXT” respectfully. They are used to create and write output data to server tier, but as this is a Web App, the user should be prompted for where and how to write the files.
3. Start Process:

* Delete data from tables CT\_EXT\_DATA, CT\_EXT\_ERRORS and CT\_EXT\_LOGS using the following code:

delete from CT\_EXT\_DATA where PROTOCOL = :CG$CTRL.STUDY

delete from CT\_EXT\_ERRORS where STUDY = :CG$CTRL.STUDY

delete from CT\_EXT\_LOGS where STUDY = :CG$CTRL.STUDY

1. Clean the Results Sections (#6 and #7)
2. Start Extract:

* For Extract Types “All” and “Since last extract” only

EXTRACT\_PKG.P\_UPD\_EXTRACT\_MODE(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO, :CG$CTRL.EXTRACT\_MODE);

* Then:

EXTRACT\_PKG.P\_UPD\_TIMESTAMP(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

* Then IF REPORT TO = CDUS

EXTRACT\_PKG.P\_BATCH\_EXTRACT(:CG$CTRL.STUDY, T\_Hold\_CDUS);

* ELSE

EXTRACT\_PKG.P\_BATCH\_EXTRACT(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

1. Update Results Sections:

* Status (#5,#6[summary]): One of Three types of Status using the following queries to update fields :CG$CTRL.RUN\_STATUS\_TEXT and :CG$CTRL.LOG\_TEXT:

In Process:

SELECT 'Extract Status: IN PROGRESS ['||FILE\_ID||' - '||PROC\_NAME||']',

'Create Extract: Processing...'

into :CG$CTRL.LOG\_TEXT,

:CG$CTRL.RUN\_STATUS\_TEXT

FROM CT\_EXT\_LOGS

WHERE STUDY = :CG$CTRL.STUDY

and EXT\_STATUS = 'PROCESSING'

and ROWNUM = 1;

Failure Found

SELECT 'Extract Status: FAIL', 'Create Extract: Completed'

into :CG$CTRL.LOG\_TEXT,

:CG$CTRL.RUN\_STATUS\_TEXT

FROM CT\_EXT\_LOGS

WHERE STUDY = :CG$CTRL.STUDY

and EXT\_STATUS = 'FAIL'

AND ROWNUM = 1;

Success Found

SELECT 'Extract Status: SUCCESS', 'Create Extract: Completed'

into :CG$CTRL.LOG\_TEXT,

:CG$CTRL.RUN\_STATUS\_TEXT

FROM CT\_EXT\_LOGS

WHERE STUDY = :CG$CTRL.STUDY

and FILE\_ID = 'ALL'

AND PROC\_NAME = 'ALL'

AND EXT\_STATUS = 'SUCCESS'

AND ROWNUM = 1;

* Extract Status Detail (#6)

Guessing that the Detail Section is a simple query that select data from CT\_EXT\_LOGS where study = :CG$CTRL.STUDY ORDER BY Study, LOG\_ID

* Errors (#7)

Guessing that this Detail Section is a simple query that selects data from CT\_EXT\_ERRORS where study = :CG$CTRL.STUDY ORDER BY study, FILE\_ID desc

1. Refresh Button (#8)

The refresh button is used to check the status of the submitted batch Job that is cause when the user selects the “Create Extract” button.

#### Generate Text Files Button

The Generate Text Files Button is used to create data files for the data extracted. It will create one physical file for each extract “File” of the extraction set. This is the secondary function of this application.

1. Only fires if Study/Report To/Version are entered.
2. Calculates Last Extract and Last Generated Dates using Extraction Package functions:

EXTRACT\_PKG.F\_LAST\_EXTRACTED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO, :CG$CTRL.VERSION);

EXTRACT\_PKG.F\_LAST\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO, :CG$CTRL.VERSION, T\_LAST\_EXTRACTED);

1. Warns if previously generated, allows user to proceed or cancel file creation.
2. Warns if data is not found in the Archive Data Set. Data must be moved to the archive tables prior to data file creation. See “Transfer To Archive Table” DO NOT PROCEED IF THIS HAPPENS
3. Create Files:

NOTE: Files were deleted previously. This will not be needed as they will be written to the HTML Server Tier first, then download. May need to have button execute, then show the files created with selectors and buttons to allow for download.

* Theradex Files:

Theradex needs one file for each file type defined for the study. It purposely does NOT create a DT file. File naming follows this template: File\_ID||Protocol||version||timstamp

* + For every file type in the studies Theradex Definition (except “DT” file types):

SELECT DISTINCT FILE\_ID, PROTOCOL, VERSION

FROM CT\_DATA

where protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') = to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

* + Get the Data Generated and put it in the file

SELECT LINE\_TEXT

FROM CT\_DATA

WHERE FILE\_ID = T\_FILE.FILE\_ID

and protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') = to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

AND SUBMISSION\_FLAG = 1

* + Update the data with the Generated Date

EXTRACT\_PKG.P\_UPD\_DATE\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO,

:CG$CTRL.VERSION, T\_LAST\_EXTRACTED, SYSDATE);

* CDUS Files:

There is only one file for CDUS, File name is built using the following template:

‘CDUS\_’||ct\_ext\_study\_rpt\_ctl.report\_as||’\_’||timestamp

* + Get the Data Generated and put it in the file

SELECT LINE\_TEXT

FROM CT\_DATA

WHERE FILE\_ID = T\_FILE.FILE\_ID

and protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') = to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

AND SUBMISSION\_FLAG = 1

* + Update the data with the Generated Date

EXTRACT\_PKG.P\_UPD\_DATE\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO,

:CG$CTRL.VERSION, T\_LAST\_EXTRACTED, SYSDATE);

#### Transfer to Archive Table Button

The Transfer to Archive Table Button is used to transfer data from the extraction processing table to the Archive Data table which holds data from all previous extracts. This action must be performed BEFORE data can be written to file, but AFTER Create Extract.

1. Only fires if Study/Report To/Version are entered.
2. The removal is performed with the following package procedure:

EXTRACT\_PKG.P\_REMOVE(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

1. :CG$CTRL.RUN\_STATUS\_TEXT is updated before and after processing.

#### Remove Last Extract Button

The Remove Last Extract Table Button is used to remove the most recent extraction data set from the Archive Table.

1. Only fires if Study/Report To/Version are entered.
2. Calculates Last Extract and Last Generated Dates using queries:

SELECT MAX(EXTRACTED) INTO T\_ARC\_DATE FROM CT\_DATA WHERE PROTOCOL = :CG$CTRL.STUDY;

SELECT MAX(CURRENT\_EXT\_DATE) INTO T\_EXT\_DATE FROM CT\_EXT\_FILE\_CTL WHERE OC\_STUDY = :CG$CTRL.STUDY;

NOTE: This needs checked as there are Package Functions that are similar:

1. If the above two dates match, and error is displayed and the transfer stops.
2. The transfer is performed with the following package procedure:

EXTRACT\_PKG.P\_TRANSFER(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

1. :CG$CTRL.RUN\_STATUS\_TEXT is updated before and after processing.

# Status (#5)

The Status field (:CG$CTRL.RUN\_STATUS\_TEXT) is used to show the current/most recent execution status of a batch execution or other button function. It is updated when any of the Action Buttons are pressed or when the Refresh Button is pressed, as those button call for a refresh of the Status Detail Sections

# Extract Status (#6)

This display table is tied to the database table CT\_EXT\_LOGS and is used to display the current execution results for the study/report to/version. Expanded details can be displayed by selecting a row within the result set. The expanded view show the entire record of CT\_EXT\_LOGS

Primary displays fields: FILE\_ID, PROC\_NAME, EXT\_STATUS, LAST\_EXT\_DATE, STUDY (ND), ERR\_ID (ND)

Detailed display fields: All Fields. Query of details based on STUDY and ERR\_ID from Primary Field. See Extract Status Details (#9)

The field :CG$CTRL.LOG\_TEXT is updated based upon the overall status of the extraction. This is done during the “Create Extraction” process. See section 8 of “Create Extraction” button.

# Error Status (#7)

This display table is tied to the database table CT\_EXT\_ERRORS and is used to display the current execution results for the study/report to/version.

Primary displays fields: FILE\_ID, CT\_ERROR\_DESC, STUDY (ND), ERR\_ID (ND)

Detailed display fields: All Fields. Query of details based on STUDY and ERR\_ID from Primary Field. See Error Status Details (#10)

The field :CG$CTRL.ERR\_TEXT is updated based upon the overall data records in this section. It counts the Total number of records and displays the count. If not error are encountered then “No Errors” is displayed.

# Refresh Button (#8)

This button is used to refresh the status sections (Extract Status and Errors) as well as the Status Text Fields :CG$CTRL.RUN\_STATUS\_TEXT, :CG$CTRL.LOG\_TEXT and CG$CTRL.ERR\_TEXT.

# Extract Status Detail (#9)

No Screen Shot. This screen displays the entire record detail of the table CT\_EXT\_LOGS when the user selects the record from the Extract Status section. Detail screen has a BACK button. Data is queried based on Study and Err\_id fields.

# Error Status Detail (#10)

No Screen Shot. This screen displays the entire record detail of the table CT\_EXT\_ERRORS when the user selects the record from the Error Status section. Detail screen has a BACK button. Data is queried based on Study and Err\_id fields.

# Security Detail (#10)

No Screen Shot. This screen displays the entire record detail of the table CT\_EXT\_ACCOUNTS which hold the list of valid user ids for the application. The screen allows for Adding and Deleting of records.

# TRIGGER CODE

#### CREATE EXTRACT

DECLARE

T\_MOD VARCHAR2(60) := 'Create Extract: ';

T\_JOB BINARY\_INTEGER;

T\_WHAT VARCHAR2(2000);

T\_DDL VARCHAR2(2000);

alert\_button NUMBER;

T\_ALERT\_MSG VARCHAR2(200);

T\_ALERT\_TITLE VARCHAR2(200);

T\_ARC\_DATE DATE;

T\_EXT\_DATE DATE;

T\_Hold\_CDUS Varchar2(30);

BEGIN

------------

IF RTRIM(:CG$CTRL.STUDY) IS NULL OR RTRIM(:CG$CTRL.REPORT\_TO) IS NULL THEN

MESSAGE(' Please select a Study and the Report\_to. ', ACKNOWLEDGE);

RETURN;

END IF;

------------

BEGIN

BEGIN

SELECT MAX(EXTRACTED) INTO T\_ARC\_DATE

FROM CT\_DATA

WHERE PROTOCOL = :CG$CTRL.STUDY;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

T\_ARC\_DATE := NULL;

END;

BEGIN

SELECT MAX(CURRENT\_EXT\_DATE) INTO T\_EXT\_DATE

FROM CT\_EXT\_FILE\_CTL

WHERE OC\_STUDY = :CG$CTRL.STUDY;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

T\_EXT\_DATE := NULL;

END;

IF T\_ARC\_DATE IS NULL OR T\_EXT\_DATE IS NULL THEN

NULL;

ELSE

IF T\_ARC\_DATE = T\_EXT\_DATE THEN

------------

T\_ALERT\_TITLE := 'Warning - This extract is already in the archive as of '||

to\_char(SYSDATE, 'MM/DD/YYYY');

T\_ALERT\_MSG := 'Extract records with cut off '||to\_char(T\_ARC\_DATE, 'MM/DD/YYYY HH:MI am')||' '||

'already exist in the archive. Continuing without removing these records will only '||

'generate records that have been updated.';

SET\_ALERT\_PROPERTY('ALERT\_EXT', ALERT\_MESSAGE\_TEXT, T\_ALERT\_MSG);

SET\_ALERT\_PROPERTY('ALERT\_EXT', TITLE, T\_ALERT\_TITLE);

alert\_button := Show\_Alert('ALERT\_EXT');

IF alert\_button = ALERT\_BUTTON2 THEN

RETURN;

END IF; -- do nothing if operator selects button 2 cancel

------------

END IF;

END IF;

IF :CG$CTRL.REPORT\_TO = 'CDUS' Then

alert\_button := Show\_Alert('ALERT\_COMP\_ABBR');

IF alert\_button = ALERT\_BUTTON1 Then

T\_Hold\_CDUS := 'CDUS-FULL';

ElsIf alert\_button = ALERT\_BUTTON2 Then

T\_Hold\_CDUS := 'CDUS-ABBR';

Else

RETURN; -- User HIT CANCEL

End If;

End If;

END;

------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' IN PROGRESS';

Synchronize;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_SUBMIT',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_SUBMIT',ENABLED,PROPERTY\_FALSE);

END IF;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED,PROPERTY\_FALSE);

END IF;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED,PROPERTY\_FALSE);

END IF;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED,PROPERTY\_FALSE);

END IF;

--------------------------------------------

-- CLEAR STAGE AND LOG TABLE----

BEGIN

T\_MOD := 'Deleting Old Candidate/Errors/Logs';

Forms\_DDL('delete from CT\_EXT\_DATA where PROTOCOL = '''||:CG$CTRL.STUDY||'''');

Forms\_DDL('delete from CT\_EXT\_ERRORS where STUDY = '''||:CG$CTRL.STUDY||'''');

Forms\_DDL('delete from CT\_EXT\_LOGS where STUDY = '''||:CG$CTRL.STUDY||'''');

GO\_BLOCK('EXTLOG');

EXECUTE\_QUERY;

GO\_BLOCK('EXTERR');

EXECUTE\_QUERY;

GO\_BLOCK('TAPDIR');

IF NOT Form\_Success THEN

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' ERROR';

ELSE

NULL;

END IF;

END;

-- EXTRACT DATA----

BEGIN

-- UPDATE EXTRACT TIMESTAMP----

IF RTRIM(:CG$CTRL.EXTRACT\_MODE) <> 'BY FILE' THEN

T\_MOD := 'P\_UPD\_EXTRACT\_MODE';

EXTRACT\_PKG.P\_UPD\_EXTRACT\_MODE(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO, :CG$CTRL.EXTRACT\_MODE);

END IF;

T\_MOD := 'P\_UPD\_TIMESTAMP';

EXTRACT\_PKG.P\_UPD\_TIMESTAMP(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

T\_MOD := 'P\_BATCH\_EXTRACT';

If :CG$CTRL.REPORT\_TO = 'CDUS' Then

EXTRACT\_PKG.P\_BATC H\_EXTRACT(:CG$CTRL.STUDY, T\_Hold\_CDUS);

Else

EXTRACT\_PKG.P\_BATCH\_EXTRACT(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

End If;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END;

----------------------------------------------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' Submission Successful';

GO\_BLOCK('EXTLOG');

EXECUTE\_QUERY;

GO\_BLOCK('EXTERR');

EXECUTE\_QUERY;

GO\_BLOCK('TAPDIR');

EXCEPTION

WHEN OTHERS THEN

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' Submission Failed';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_SUBMIT',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_SUBMIT',ENABLED,PROPERTY\_TRUE);

END IF;

RAISE;

END;

#### TRANSFER TO ARCHIVE TABLE

DECLARE

T\_MOD VARCHAR2(60) := 'Transfer data to the archive table: ';

T\_ARC\_DATE DATE;

T\_EXT\_DATE DATE;

BEGIN

------------

IF RTRIM(:CG$CTRL.STUDY) IS NULL OR RTRIM(:CG$CTRL.REPORT\_TO) IS NULL THEN

MESSAGE(' Please select a Study and the Report\_to. ', ACKNOWLEDGE);

RETURN;

END IF;

------------

BEGIN

BEGIN

SELECT MAX(EXTRACTED) INTO T\_ARC\_DATE

FROM CT\_DATA

WHERE PROTOCOL = :CG$CTRL.STUDY;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

T\_ARC\_DATE := NULL;

END;

BEGIN

SELECT MAX(CURRENT\_EXT\_DATE) INTO T\_EXT\_DATE

FROM CT\_EXT\_FILE\_CTL

WHERE OC\_STUDY = :CG$CTRL.STUDY;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

T\_EXT\_DATE := NULL;

END;

IF T\_ARC\_DATE IS NULL OR T\_EXT\_DATE IS NULL THEN

NULL;

ELSE

IF T\_ARC\_DATE = T\_EXT\_DATE THEN

MESSAGE('This Extract is already in the Archive.', ACKNOWLEDGE);

MESSAGE(' ');

RETURN;

END IF;

END IF;

END;

------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' IN PROGRESS';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED,PROPERTY\_FALSE);

END IF;

-------------------------

EXTRACT\_PKG.P\_TRANSFER(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO); -- prc 10/21/03

-------------------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' SUCCESS';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED,PROPERTY\_TRUE);

END IF;

EXCEPTION

WHEN OTHERS THEN

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' FAIL';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TRANSFER',ENABLED,PROPERTY\_TRUE);

END IF;

RAISE;

END;

#### GENERATE TEXT FILES

DECLARE

T\_MOD VARCHAR2(60) := 'Generate TAP files: ';

T\_out\_file Text\_IO.File\_Type;

T\_FILE\_NAME VARCHAR2(120) := NULL;

T\_LINE\_TEXT VARCHAR2(2400) := NULL;

T\_FILEDIR VARCHAR2(60) := NULL;

T\_FILEEXT VARCHAR2(60) := NULL;

T\_TIMESTAMP VARCHAR2(60) := NULL;

T\_LAST\_EXTRACTED DATE;

T\_LAST\_GENERATED DATE;

alert\_button NUMBER;

T\_ALERT\_MSG VARCHAR2(200);

T\_ALERT\_TITLE VARCHAR2(200);

BEGIN

------------

IF RTRIM(:CG$CTRL.STUDY) IS NULL OR RTRIM(:CG$CTRL.REPORT\_TO) IS NULL THEN

MESSAGE(' Please select a Study and the Report\_to. ', ACKNOWLEDGE);

RETURN;

END IF;

------------

BEGIN

T\_LAST\_EXTRACTED := EXTRACT\_PKG.F\_LAST\_EXTRACTED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO, :CG$CTRL.VERSION);

T\_LAST\_GENERATED := EXTRACT\_PKG.F\_LAST\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO,

:CG$CTRL.VERSION, T\_LAST\_EXTRACTED);

IF (T\_LAST\_GENERATED IS NOT NULL) THEN

------------

T\_ALERT\_TITLE := 'Warning - The '||:CG$CTRL.REPORT\_TO||' text files have been generated.';

T\_ALERT\_MSG := ' The text files with cutoff date '||to\_char(T\_LAST\_EXTRACTED, 'MM/DD/YYYY')||

' have been generated from archive for '||:CG$CTRL.STUDY||

' on '||to\_char(T\_LAST\_GENERATED, 'MM/DD/YYYY')||'. '||

'Continuing will generate the text files again. ';

SET\_ALERT\_PROPERTY('ALERT\_GEN', ALERT\_MESSAGE\_TEXT, T\_ALERT\_MSG);

SET\_ALERT\_PROPERTY('ALERT\_GEN', TITLE, T\_ALERT\_TITLE);

alert\_button := Show\_Alert('ALERT\_GEN');

IF alert\_button = ALERT\_BUTTON2 THEN

RETURN;

END IF; -- do nothing if operator selects button 2 cancel

------------

ELSE

IF (T\_LAST\_EXTRACTED IS NOT NULL) THEN

------------

T\_ALERT\_TITLE := :CG$CTRL.REPORT\_TO||' text files ';

T\_ALERT\_MSG := 'You are about to generate text files with cutoff date '||

to\_char(T\_LAST\_EXTRACTED, 'MM/DD/YYYY')||' from archive for '||:CG$CTRL.STUDY;

SET\_ALERT\_PROPERTY('ALERT\_INFO1', ALERT\_MESSAGE\_TEXT, T\_ALERT\_MSG);

SET\_ALERT\_PROPERTY('ALERT\_INFO1', TITLE, T\_ALERT\_TITLE);

alert\_button := Show\_Alert('ALERT\_INFO1');

IF alert\_button = ALERT\_BUTTON2 THEN

RETURN;

END IF; -- do nothing if operator selects button 2 cancel

------------

ELSE

------------

T\_ALERT\_TITLE := :CG$CTRL.REPORT\_TO||' text files ';

T\_ALERT\_MSG := 'The extract with cutoff date '||to\_char(T\_LAST\_EXTRACTED, 'MM/DD/YYYY')||

' for '||:CG$CTRL.STUDY||' doesn''t exist in the archive. ';

SET\_ALERT\_PROPERTY('ALERT\_INFO2', ALERT\_MESSAGE\_TEXT, T\_ALERT\_MSG);

SET\_ALERT\_PROPERTY('ALERT\_INFO2', TITLE, T\_ALERT\_TITLE);

alert\_button := Show\_Alert('ALERT\_INFO2');

IF alert\_button = ALERT\_BUTTON1 THEN

RETURN;

END IF; -- do nothing

------------

END IF;

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE;

END;

--------------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' IN PROGRESS';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED,PROPERTY\_FALSE);

END IF;

------------------------

----DELETE FILES-----------------------------

IF :SYSTEM.BLOCK\_STATUS = 'CHANGED' THEN

COMMIT;

END IF;

SELECT EXT\_UTIL\_PKG.F\_APP\_META\_DATA('FILEDIR')

,EXT\_UTIL\_PKG.F\_APP\_META\_DATA('FILEEXT')

,TO\_CHAR(SYSDATE, 'RRRRMMDD')

INTO T\_FILEDIR, T\_FILEEXT, T\_TIMESTAMP

FROM DUAL;

-- T\_FILE\_NAME := T\_FILEDIR||'\*'||T\_FILEEXT;

BEGIN

HOST('MKDIR '||SUBSTR(T\_FILEDIR, 1, LENGTH(T\_FILEDIR) - 1));

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

----------------------

IF :CG$CTRL.REPORT\_TO = 'THERADEX' THEN

--------------------

--Message('IF :CG$CTRL.REPORT\_ TO = ''THERADEX''');

FOR T\_FILE IN (SELECT PROTOCOL, VERSION

FROM CT\_DATA

where protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

GROUP BY PROTOCOL, VERSION ) LOOP

T\_FILE\_NAME := T\_FILEDIR||'\*'||T\_FILE.PROTOCOL||T\_FILE.VERSION||T\_TIMESTAMP||T\_FILEEXT;

BEGIN

HOST('DEL '||T\_FILE\_NAME);

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

END LOOP;

----------------

BEGIN

FOR T\_FILE IN (SELECT DISTINCT FILE\_ID, PROTOCOL, VERSION

FROM CT\_DATA

where protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') = to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')) LOOP

IF T\_FILE.FILE\_ID <> 'DT' THEN

T\_FILE\_NAME := T\_FILEDIR||T\_FILE.FILE\_ID||T\_FILE.PROTOCOL||T\_FILE.VERSION||T\_TIMESTAMP||T\_FILEEXT;

--Message('File = "'||T\_FILE\_NAME||'".');

T\_out\_file := Text\_IO.Fopen(T\_FILE\_NAME, 'w');

/\* PRC : 03/30/2005: "PH" special processing removed.

IF T\_FILE.FILE\_ID <> 'PH' THEN

FOR T\_REC IN (SELECT LINE\_TEXT

FROM CT\_DATA

WHERE FILE\_ID = T\_FILE.FILE\_ID

and protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

AND SUBMISSION\_FLAG = 1) LOOP

Text\_IO.Put\_Line(T\_out\_file, T\_REC.LINE\_TEXT);

END LOOP;

ELSE

FOR T\_REC IN (SELECT EXT\_UTIL\_PKG.F\_PH\_TEXT\_1(PROTOCOL,PATIENT,DEL\_FLAG,NOTE\_DT,FILE\_ID,TYP\_CD) LINE\_TEXT

FROM (SELECT PROTOCOL,PATIENT,DEL\_FLAG,KEY1 NOTE\_DT,KEY2 FILE\_ID, KEY3 TYP\_CD

FROM CT\_DATA

WHERE FILE\_ID = T\_FILE.FILE\_ID

and protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

AND SUBMISSION\_FLAG = 1

GROUP BY PROTOCOL,PATIENT,DEL\_FLAG,KEY1,KEY2,KEY3) ) LOOP

--Message(T\_REC.LINE\_TEXT);

Text\_IO.Put\_Line(T\_out\_file, T\_REC.LINE\_TEXT);

END LOOP;

END IF;

\*\*\*\*\*\*/

FOR T\_REC IN (SELECT LINE\_TEXT

FROM CT\_DATA

WHERE FILE\_ID = T\_FILE.FILE\_ID

and protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

AND SUBMISSION\_FLAG = 1) LOOP

Text\_IO.Put\_Line(T\_out\_file, T\_REC.LINE\_TEXT);

END LOOP;

Text\_IO.Fclose (T\_out\_file);

END IF;

END LOOP;

--------------------

BEGIN

EXTRACT\_PKG.P\_UPD\_DATE\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO,

:CG$CTRL.VERSION, T\_LAST\_EXTRACTED, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

RAISE;

END;

--------------------

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED,PROPERTY\_TRUE);

END IF;

EXCEPTION

WHEN OTHERS THEN

IF Text\_IO.Is\_Open(T\_out\_file) THEN

Text\_IO.Fclose(T\_out\_file);

END IF;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABL ED,PROPERTY\_TRUE);

END IF;

RAISE;

END;

--------------------

ELSIF :CG$CTRL.REPORT\_TO = 'CDUS' THEN

--------------------

FOR T\_FILE IN (SELECT PROTOCOL, VERSION

FROM CDUS\_DATA

where protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

GROUP BY PROTOCOL, VERSION ) LOOP

T\_FILE\_NAME := T\_FILEDIR||'\*'||T\_FILE.PROTOCOL||T\_FILE.VERSION||T\_TIMESTAMP||T\_FILEEXT;

BEGIN

HOST('DEL '||T\_FILE\_NAME);

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

END LOOP;

----------------

BEGIN

-- prc 02/24/2006:

-- Added specific "T\_TIMESTAMP" build for CDUS. Added HH24.

SELECT EXT\_UTIL\_PKG.F\_APP\_META\_DATA('FILEDIR')

,EXT\_UTIL\_PKG.F\_APP\_META\_DATA('FILEEXT')

,TO\_CHAR(SYSDATE, 'RRRRMMDDHH24')

INTO T\_FILEDIR, T\_FILEEXT, T\_TIMESTAMP

FROM DUAL;

/\*FOR T\_FILE IN (SELECT PROTOCOL, VERSION

FROM CDUS\_DATA

WHERE protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

and ROWNUM = 1) LOOP \*/

-- prc 02/24/2006

-- Replace for..loop above with one below. Gets "report\_as" value instead of study to build file name

-- with a more identifiable name.

For T\_File in (SELECT REPORT\_TO, VERSION, REPORT\_AS

FROM CT\_EXT\_STUDY\_RPT\_CTL

WHERE OC\_STUDY = RTRIM(:CG$CTRL.STUDY)

AND REPORT\_TO = :CG$CTRL.REPORT\_TO

AND ROWNUM = 1) Loop

T\_FILE\_NAME := T\_FILEDIR||'CDUS\_'||T\_FILE.REPORT\_AS||'\_'||T\_TIMESTAMP||T\_FILEEXT;

T\_out\_file := Text\_IO.Fopen(T\_FILE\_NAME, 'w');

-- prc 02/24/2006: changed order by to include "PATIENT, KEY1, KEY2, KEY3, KEY4, KEY5"

FOR T\_REC IN (SELECT LINE\_TEXT

FROM CDUS\_DATA a,

CT\_EXT\_CRS\_CTL b

WHERE protocol = RTRIM(:CG$CTRL.STUDY)

and to\_date(to\_char(extracted,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss') =

to\_date(to\_char( T\_LAST\_EXTRACTED ,'yyyy/mm/dd hh24:mi:ss'),'yyyy%mm%dd hh24:mi:ss')

and a.PROTOCOL = b.OC\_STUDY

and a.FILE\_ID = b.file\_id

and a.VERSION = b.VERSION

order by b.ext\_seq, PATIENT, KEY1, KEY2, KEY3, KEY4, KEY5

) LOOP

Text\_IO.Put\_Line(T\_out\_file, T\_REC.LINE\_TEXT);

END LOOP;

Text\_IO.Fclose (T\_out\_file);

END LOOP;

--------------------

BEGIN

EXTRACT\_PKG.P\_UPD\_DATE\_GENERATED(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO,

:CG$CTRL.VERSION, T\_LAST\_EXTRACTED, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

RAISE;

END;

--------------------

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED,PROPERTY\_TRUE);

END IF;

EXCEPTION

WHEN OTHERS THEN

IF Text\_IO.Is\_Open(T\_out\_file) THEN

Text\_IO.Fclose(T\_out\_file);

END IF;

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_TAP',ENABLED,PROPERTY\_TRUE);

END IF;

RAISE;

END;

--------------------

ELSE

NULL;

END IF;

------------------------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' SUCCESS';

EXCEPTION

WHEN OTHERS THEN

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' FAIL'||SubStr(sqlERRM,1,100);

RAISE;

END;

#### REMOVE LAST EXTRACT

DECLARE

T\_MOD VARCHAR2(60) := 'Remove the Last Extract: ';

BEGIN

------------

IF RTRIM(:CG$CTRL.STUDY) IS NULL OR RTRIM(:CG$CTRL.REPORT\_TO) IS NULL THEN

MESSAGE(' Please select a Study and the Report\_to. ', ACKNOWLEDGE);

RETURN;

END IF;

------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' IN PROGRESS';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED) = 'TRUE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED,PROPERTY\_FALSE);

END IF;

-------------------------

EXTRACT\_PKG.P\_REMOVE(:CG$CTRL.STUDY, :CG$CTRL.REPORT\_TO);

-------------------------

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' SUCCESS';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED,PROPERTY\_TRUE);

END IF;

EXCEPTION

WHEN OTHERS THEN

:CG$CTRL.RUN\_STATUS\_TEXT := T\_MOD||' FAIL';

IF GET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED) = 'FALSE' THEN

SET\_ITEM\_PROPERTY('CG$CTRL.BUT\_REMOVE',ENABLED,PROPERTY\_TRUE);

END IF;

RAISE;

END;