**16 October 2018**

DMHRSi HR Basic File

for the

MHS Data Repository (MDR)

(Version 1.3.0)

Future Specification

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Para/Tbl/Fig** | **Originator** | **Description of Change** |
| 1.0.0 | 04/10/2012 | Whole document | H. Escobar | Baseline Document |
| 1.0.1 | 05/02/2012 | Table 1 | H. Escobar | Revised Person\_Type and Service variable lengths |
| 1.1.0 | 04/07/2015 | Sections I, II, III  | H. Escobar | Update ICD reference; remove requirement for annual full data refresh and weekly raw data updates |
| 1.2.0 | 3/17/2016 | Section V, Table 1 | H. Escobar | Added creation of Derived\_DODOCC through lookup by SUOC against EASIV SUOC/DODOCC table. |
| 1.3.0 | 10/16/2018 | Sections I | C. McCrocklin | Update ICD reference; Updated Appendix A: Incremental Extract SQL Code |

**MDR DMHRSi-HR Basic File**

1. Source

Data capture system: Defense Medical Human Resources System (internet) (DMHRSi) documented in **SDDDMHRSiMDRICDFinal30 August 2018v214.docx**.

1. Transmission (Format and Frequency)

The initial file load is a one-time requirement containing selected DMHRSi-HR data from 1 Oct 2008 (FY09 Start) to 30 Sept 2011 (FY11 End). Subsequently, DHSS will provide monthly raw ASCII text “!” delimited DMHRSi HR data extracts via FTP as described in the interface control document (ICD) referenced above. The incremental extracts reflect updated or added DMHRSi personnel records since the most recent extract (see Appendix A for incremental extract SQL code). DHSS staff will provide the following DMHRSi HR extract:

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| DMHRSi-HR Extract(Table A-1: dmhrsi.txt) | Contains incrementally updated DMHRSi-HR raw data since most recent extract |

1. Organization and batching
2. DMHRSi extracts are received and processed monthly.

1. After processing, each updated DMHRSi-HR Basic file is saved as individual SAS datasets containing DMHRSi-HR data from 1 Oct 2008 to extract date of the most recent incremental file.
2. Receiving Filters
3. Processor should scan each incoming raw incremental update and remove non-visible characters imbedded in the raw data observations (i.e., tabs, line breaks, etc.).
4. Filter and correct erroneous delimiter characters “!” imbedded inside valid observations (i.e., "KV!MFV9P" vs. "KV1MFV9P").
5. Processor should remove duplicate records that may be included in the raw files.
6. Field Transformations and Deletions for MDR Core Database
7. The existing master DMHRSi-HR Basic file will be updated monthly with incremental data sets. The Processor should rely on a combination of “Person ID” and “Assignment Effective Start Date” to assess whether person records received in the incremental files are present in the master file. If so, the specific record should be replaced with the content in the incremental DMHRSi-HR extract. If there is no “Person ID” and “Assignment Effective Start Date” match to the existing DMHRSi-HR dataset, the content in the incremental DMHRSi-HR extract should be appended to the master DMHRSi-HR dataset.
8. The Processor will note the processing date personnel records contained in the incremental data input files are added using the PROCDATE field as “MM/DD/YYYY”.
9. All dates should use SAS format “MMDDYY10.”.
10. A new variable “Derived\_DODOCC” will be created in the DMHRSi HR SAS file during monthly processing by merging against the most recent SUOC/DODOCC reference file available by SUOC and SKILL\_TYPE. Presently, this reference file is /mdr/aref/dmhrsi/d160316/occmap.sas7bdat. As map updates occur, newer files will be posted in /mdr/aref/dmhrsi/. Updates will be made at least annually, or as needed. All observations in the source DMHRSi HR SAS dataset should be retained, and only matching observations in the reference file should be kept. In addition, observations in the post-merged DMHRSi HR SAS dataset where Skill\_Type is equal to “5” and SUOC is in the range "0","00","000","0000","00000","0000E","0000O","0000C", the new “Derived\_DODOCC” variable should be set to “NONMED”. Below is sample SAS code for this step where SAS dataset “one” is the DMHRSi HR SAS dataset and SAS dataset “two” is the most recent SUOC/DODOCC reference file:

**proc** **sort** data=one;

by suoc skill\_type; **run**;

**proc** **sort** data=two;

by suoc skill\_type; **run**;

**data** comb;

merge one(in=dat) two;

by suoc skill\_type;

if dat;

if skill\_type ="5" and suoc in ("0","00","000","0000","00000","0000E","0000O","0000C") then derived\_dodocc = "NONMED";

run;

1. File layout and content

The MDR DMHRSi-HR Basic file is a SAS Data Set.

**TABLE 1: MDR DMHRSi-HR Basic FILE**

| **Field Name** | **Field Length** | **Data Type** | **SAS Name** | **Functional Description** |
| --- | --- | --- | --- | --- |
| Processing Date (SAS Date) | 10 | Date | PROCDATE | Date of record processingFormat MM/DD/YYYY |
| SSN | 11 | Char | SSN | Social Security Number |
| EDIPN | 10 | Char | EDIPN | DEERS Electronic Data Interchange Person Number |
| NPI | 10 | Char | NPI | National Provider Identifier |
| Person ID | 8 | Char | PERSON\_ID | Unique DMHRSi Person id |
| Employee Number | 8 | Char | EMP\_NUM | Unique DMHRSi Person Number |
| Employee Assignment DMIS ID | 4 | Char | ASSIG\_DMISID | Employee's DMIS ID of Assignment |
| Employee People Group DMIS ID | 4 | Char | PG\_DMISID | People Group Org DMISID where labor time is recorded |
| Position ID | 8 | Char | POSITION\_ID | Unique DMHRSi Position id |
| Job ID | 8 | Char | JOB\_ID | Unique DMHRSi Job id |
| Organization ID | 8 | Char | ORG\_ID | Unique DMHRSi Organization ID |
| People Group ID | 8 | Char | PG\_ID | Unique DMHRSi People Group ID |
| Assignment Effective Start Date (SAS Date) | 10 | Date | ASSIG\_START | Employee's Assignment Start DateFormat MM/DD/YYYY |
| Assignment Effective End Date (SAS Date) | 10 | Date | ASSIG\_END | Employee's Assignment End DateFormat MM/DD/YYYY |
| Person Effective Start Date (SAS Date) | 10 | Date | PERSON\_START | HR Record Start DateFormat MM/DD/YYYY |
| Person Effective End Date (SAS Date) | 10 | Date | PERSON\_END | HR Record End DateFormat MM/DD/YYYY |
| Original Date of Hire (SAS Date) | 10 | Date | HIRE\_DATE | Employee's Original Date of HireFormat MM/DD/YYYY |
| Assignment DMIS ID Service | 9 | Char | ASSIG\_SERVICE | Branch of Service of Assignment DMISID |
| People Group DMIS ID Service | 9 | Char | PG\_SERVICE | Service Branch of People Group DMISID where time is recorded |
| Last Name | 28 | Char | LASTNAME | Employee last name |
| First Name | 28 | Char | FIRSTNAME | Employee first name |
| Middle Name | 20 | Char | MIDNAME | Employee middle name |
| Suffix | 4 | Char | SUFFIX | Employee Name Suffix |
| Title/Rank | 12 | Char | TITLE\_RANK | Employee's Title or Rank |
| Person Grade | 8 | Char | GRADE | Civilian/Active Duty Grade |
| Person Step | 2 | Char | STEP | Employee's Step associated with grade |
| Gender | 1 | Char | GENDER | Employee Gender |
| Person Type | 22 | Char | PERSON\_TYPE | Employee Personnel Category |
| Skill Type | 1 | Char | SKILL\_TYPE | Person Skill Type |
| Skill Type Suffix | 1 | Char | SKILL\_SUFFIX | Skill Type Suffix Code |
| Organization UIC  | 8 | Char | ORG\_UIC | UIC Associated with employee's organization |
| People Group UIC | 8 | Char | PG\_UIC | UIC Associated with employee's People Group |
| DOD Occupation Code | 10 | Char | DODOCC | DoD Occupation Code associated with job |
| Taxonomy Code 1 | 11 | Char | TAX1 | HIPAA Taxonomy code 1 |
| Taxonomy Code 2 | 11 | Char | TAX2 | HIPAA Taxonomy code 2 |
| Taxonomy Code 3 | 11 | Char | TAX3 | HIPAA Taxonomy code 3 |
| Taxonomy Code 4 | 11 | Char | TAX4 | HIPAA Taxonomy code 4 |
| Taxonomy Code 5 | 11 | Char | TAX5 | HIPAA Taxonomy code 5 |
| Taxonomy Code 6 | 11 | Char | TAX6 | HIPAA Taxonomy code 6 |
| Person Service | 14 | Char | SERVICE | Person's Branch of Service  |
| Person UIC/PAS | 8 | Char | UIC | UIC Associated with employee |
| SUOC | 8 | Char | SUOC | Service-Unique Occupation Code |
| Record ID | 24 | Char | RECORD\_ID | Unique DMHRSi Record Id |
| Creation Date (SAS Date) | 10 | Date | RECORD\_DATE | Date DMHRSi Record was createdFormat MM/DD/YYYY |
| Last Update Date (SAS Date) | 10 | Date | UPDATE\_DATE | Date DMHRSi Record was last updatedFormat MM/DD/YYYY |
| People Group FCC | 4 | Char | PG\_ASSIG\_FCC | Employee's People Group Assignment FCC |
| Derived DoD Occupation Code from EASIV Table | 6 | Char | DERIVED\_DODOCC | Derived DoD Occupation Code from EASIV SUOC to DODOCC Table |

1. Refresh Frequency

The DMHRSi-HR Basic file will be updated monthly with new or changed personnel records since the most recent harvest date.

1. Data Marts

None.

1. Special Outputs

None.

1. Quality assurance

The processor should conduct quality assurance checks on every processed output file to ensure input and output data are valid, complete, and reliable. At a minimum, the processor should:

* Ensure no “Lost Cards” result during raw text import into SAS. This is common if random hex character commands are imbedded in valid observations.
* Compare monthly raw data row counts to ensure changed/added records match raw input row counts.
* Evaluate post-processing values for data that appear out of the ordinary, or not consistent with SME expected values (face validity).

Appendix A: Incremental Extract SQL Code

(Edit Begin/End Incremental Dates as Appropriate)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE OR REPLACE PACKAGE dmhrsi.dod\_cber\_mdr\_extract

AUTHID CURRENT\_USER IS

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Package Name: dod\_cber\_mdr\_extract

-- Description:

--

-- Modification Log for Comments and Description

-- Date Comments

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

-- 7-Feb-2017 Richard Koonce

-- Initial development, converted from anonymous block.

-- CR-16420 Title: Convert file for DMHRSi to MDR Data Feed

-- Provides personnel information to MDR.

-- Only Assignments active Oct. 1, 2011 and after.

-- Pulls data active or updated during prior month

-- - Assign active (starting) on or before end of prior month

-- assgn start date < first of this month (vc\_current\_month\_start\_dt)

-- Procedure dmhrsi.dod\_cber\_extract created in February 2017

--

-- 7-Mar-2018 Richard Koonce

-- CR19212-Modify the feed from The Defense Medical Human

-- Resources System-Internet (DMHRSi) to the MHS Data Repository

-- (MDR) to allow for full refresh of data.

-- Created package from previous procedure dmhrsi.dod\_cber\_extract.

-- So extract file can be moved from concurrent manager output directory

-- to file name and directory path specified in parameters.

-- Additional changes and information in body header.

--

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Main routine to control processing

PROCEDURE extract\_process\_control

 ( errbuf OUT VARCHAR2

 , retcode OUT NUMBER

 , p\_start\_date\_month IN NUMBER

 , p\_start\_date\_year IN NUMBER

 , p\_target\_file\_name IN VARCHAR2

 , p\_target\_directory IN VARCHAR2

 );

-- Routine to write extract file to concurrent output

PROCEDURE generate\_extract\_file

 ( errbuf OUT VARCHAR2

 , retcode OUT NUMBER

 , p\_period\_start\_date IN DATE

 );

END dod\_cber\_mdr\_extract;

/

CREATE OR REPLACE PACKAGE BODY dmhrsi.dod\_cber\_mdr\_extract

IS

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Package Name: dod\_cber\_mdr\_extract

-- Description:

-- Package

--

--

--

-- Modification Log for Comments and Description

-- Date Comments

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

-- 7-Feb-2017 Richard Koonce

-- Initial development, converted from anonymous block.

-- See Specification header comments for details.

-- Procedure dmhrsi.dod\_cber\_extract created in February 2017

--

-- 7-Mar-2018 Richard Koonce

-- CR19212-Modify the feed from The Defense Medical Human

-- Resources System-Internet (DMHRSi) to the MHS Data Repository

-- (MDR) to allow for full refresh of data.

-- Created package from previous procedure dmhrsi.dod\_cber\_extract.

-- So extract file can be moved from concurrent manager output directory

-- to file name and directory path specified in parameters.

--

-- Added 4 input parameters, month, year, target filename and target

-- directory. Added routines from the original one procedure. The

-- process has a control routine which calls a routine to generate the

-- extract file, via concurrent submission, and submits a concurrent

-- request to copy output file to a requested filename and directory.

--

-- Changes to the generate extract file included the following.

-- Modify query to be full extract from a given start date (parameter).

-- Changed cursor from 4 inputs to just one. Include Assignment records

-- effective as of start date parameter. People, NPI (Qualification),

-- Person-Type, Organization-LCA and People-Group-LCA records are

-- included when they are effective during the Assignment record

-- (effective start to effective end dates) and ending after start date

-- parameter. Expenditure-Item-Date (effective date) used for SUOC

-- derivation is the most recent date of either the Assignment Effective

-- Start Date or the Person Duty Occupation Start Date when present.

--

-- Added routine to calculate the elapsed run time for the extract

-- and query process details from concurrent requests table, then

-- insert into an Interface Log table, via an external call to

-- procedure dmhrsi.log\_interface\_information.

--

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

-- Global Constant Declarations

gc\_debug CONSTANT BOOLEAN := TRUE;

-- CC-request submit constants and programs

gc\_cc\_wait\_interval\_sec CONSTANT NUMBER := 30; -- wait interval between checking request

gc\_cc\_max\_wait\_length\_sec CONSTANT NUMBER := 14400; -- max wait time on request (equals 4 hours)

gc\_dmhrsi\_appl\_short\_name CONSTANT applsys.fnd\_application.application\_short\_name%TYPE

 := 'CUST\_HR';

gc\_file\_copy\_cc\_prog CONSTANT applsys.fnd\_concurrent\_programs.concurrent\_program\_name%TYPE

 := 'DOD\_FILE\_COPY\_SHELL\_SCRIPT';

gc\_gen\_extract\_file\_cc\_prog CONSTANT applsys.fnd\_concurrent\_programs.concurrent\_program\_name%TYPE

 := 'DOD\_CBER\_GENERATE\_FILE';

-- Global Constants for File Processing

gc\_concurrent\_output\_dir CONSTANT VARCHAR2(20) := '$APPLCSF/$APPLOUT';

-- Concurrent Request Return Code Values

-- 0-Success, 1-Warning, 2-Failure

gc\_success\_status\_code CONSTANT NUMBER := 0;

gc\_warning\_status\_code CONSTANT NUMBER := 1;

gc\_failure\_status\_code CONSTANT NUMBER := 2;

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

--- Forward Declarations

--- Processing routines are coded in alphabetic order

--- Functions first then Procedures

--- Public routines are at end of package

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

FUNCTION concurrent\_request\_name

 ( p\_concurrent\_request\_id IN NUMBER

 ) RETURN VARCHAR2;

FUNCTION elapsed\_run\_time

 ( p\_start\_date\_time IN DATE

 , p\_end\_date\_time IN DATE

 ) RETURN VARCHAR2;

FUNCTION log\_extract\_process\_info

 ( p\_process\_request\_id IN NUMBER

 , p\_process\_control\_status IN NUMBER

 , p\_records\_in\_extract IN NUMBER

 ) RETURN BOOLEAN;

FUNCTION submit\_copy\_concurrent\_file

 ( p\_source\_request\_id IN NUMBER

 , p\_target\_dir\_path IN VARCHAR2

 , p\_target\_file\_name IN VARCHAR2

 ) RETURN BOOLEAN;

PROCEDURE message\_output

 ( p\_destination\_file IN VARCHAR2 DEFAULT 'LOG'

 , p\_message\_text IN VARCHAR2 );

PROCEDURE submit\_generate\_extract\_file

 ( p\_period\_start\_date IN DATE

 , p\_extract\_request\_id OUT NUMBER

 , p\_routine\_status\_code OUT NUMBER );

--- End Forward Declarations

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

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

-- Function concurrent\_request\_name

-- Queries concurrent requests table and returns the displayed

-- request name for the provided request-id.

-- Exceptions are ignored, as the name is not significant during processing

--

FUNCTION concurrent\_request\_name

 ( p\_concurrent\_request\_id IN NUMBER

 ) RETURN VARCHAR2 IS

 CURSOR c\_concurrent\_program\_name IS

 SELECT program

 FROM apps.fnd\_conc\_req\_summary\_v

 WHERE request\_id = p\_concurrent\_request\_id;

 v\_concurrent\_program\_name apps.fnd\_conc\_req\_summary\_v.program%TYPE;

BEGIN -- Function concurrent\_request\_name

 OPEN c\_concurrent\_program\_name;

 FETCH c\_concurrent\_program\_name INTO v\_concurrent\_program\_name;

 IF c\_concurrent\_program\_name%NOTFOUND THEN

 v\_concurrent\_program\_name := NULL;

 END IF;

 CLOSE c\_concurrent\_program\_name;

 RETURN v\_concurrent\_program\_name;

 EXCEPTION

 WHEN OTHERS THEN

 -- name is not significant so no messages

 RETURN NULL;

END concurrent\_request\_name;

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

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

FUNCTION elapsed\_run\_time

 ( p\_start\_date\_time IN DATE

 , p\_end\_date\_time IN DATE

 ) RETURN VARCHAR2 IS

 --

 -- local routine variables

 --

 v\_elapsed\_time\_hours NUMBER;

 v\_elapsed\_time\_minutes NUMBER;

 v\_elapsed\_time\_seconds NUMBER;

 v\_process\_elapsed\_time VARCHAR2(12); -- ( hours:minutes:seconds )

 v\_code\_position VARCHAR2(100);

BEGIN -- Function elapsed\_run\_time

 v\_code\_position := 'Routine Start';

 --IF gc\_debug THEN

 --message\_output('LOG', NULL);

 --message\_output('LOG', 'Calculating the Elapsed Run Time.' );

 -- message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 --END IF;

 --

 v\_code\_position := 'Convert Elapsed Time to Hours, Minutes, Seconds';

 v\_elapsed\_time\_hours := (p\_end\_date\_time - p\_start\_date\_time) \* 24;

 v\_elapsed\_time\_minutes := (v\_elapsed\_time\_hours - TRUNC(v\_elapsed\_time\_hours)) \* 60;

 v\_elapsed\_time\_seconds := ROUND((v\_elapsed\_time\_minutes - TRUNC(v\_elapsed\_time\_minutes)) \* 60);

 --

 v\_code\_position := 'Is Hours at Least 2 Characters Without Padding';

 IF LENGTH(TRUNC(v\_elapsed\_time\_hours)) < 2 THEN

 -- want format hh:mm:ss, so hours needs to be padded

 v\_process\_elapsed\_time

 := LPAD(TRUNC(v\_elapsed\_time\_hours),2,'0')

 ||':'|| LPAD(TRUNC(v\_elapsed\_time\_minutes),2,'0')

 ||':'|| LPAD(v\_elapsed\_time\_seconds,2,'0');

 ELSE

 -- hours is at least 2 digits, so have format hh:mm:ss without padding hours

 v\_process\_elapsed\_time

 := TRUNC(v\_elapsed\_time\_hours)

 ||':'|| LPAD(TRUNC(v\_elapsed\_time\_minutes),2,'0')

 ||':'|| LPAD(v\_elapsed\_time\_seconds,2,'0');

 END IF;

 --

 --IF gc\_debug THEN

 -- message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 --END IF;

 v\_code\_position := 'Calculating the Elapsed Run Time Completed';

 --

 RETURN v\_process\_elapsed\_time;

 --

 EXCEPTION

 WHEN OTHERS THEN

 message\_output('LOG', NULL);

 message\_output('LOG', ' Unexpected Error Calculating the Elapsed Run Time.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' Error after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Exit Calculating the Elapsed Run Time Routine.' );

 message\_output('LOG', NULL);

 RETURN NULL;

END elapsed\_run\_time;

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

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

-- Function log\_extract\_process\_info

-- Inserts the process information into the interface logging table.

-- Routine has to convert

-- Returns status of insert as function and writes any exceptions encountered.

--

FUNCTION log\_extract\_process\_info

 ( p\_process\_request\_id IN NUMBER

 , p\_process\_control\_status IN NUMBER

 , p\_records\_in\_extract IN NUMBER

 ) RETURN BOOLEAN IS

 --

 -- request information for extract information log table

 CURSOR c\_concurrent\_request\_info

 ( cp\_request\_id NUMBER ) IS

 SELECT requests.actual\_start\_date

 , requests.requested\_by

 , requests.responsibility\_id

 , requests.argument\_text

 , requests.concurrent\_program\_id

 , DECODE ( requests.description

 , NULL, prog\_tls.user\_concurrent\_program\_name

 , requests.description ||' ('

 || prog\_tls.user\_concurrent\_program\_name ||')'

 ) current\_program\_name

 FROM applsys.fnd\_concurrent\_requests requests

 , applsys.fnd\_concurrent\_programs\_tl prog\_tls

 WHERE requests.program\_application\_id = prog\_tls.application\_id

 AND requests.concurrent\_program\_id = prog\_tls.concurrent\_program\_id

 AND prog\_tls.language = USERENV('LANG')

 AND requests.request\_id = cp\_request\_id;

 --

 -- local routine variables and interface log info variables

 --

 v\_completion\_date DATE;

 v\_process\_elapsed\_time dmhrsi.dod\_interface\_info\_log.process\_elapsed\_time%TYPE;

 v\_request\_status dmhrsi.dod\_interface\_info\_log.request\_status%TYPE;

 v\_actual\_start\_date applsys.fnd\_concurrent\_requests.actual\_start\_date%TYPE;

 v\_requested\_by applsys.fnd\_concurrent\_requests.requested\_by%TYPE;

 v\_responsibility\_id applsys.fnd\_concurrent\_requests.responsibility\_id%TYPE;

 v\_argument\_text applsys.fnd\_concurrent\_requests.argument\_text%TYPE;

 v\_concurrent\_program\_id applsys.fnd\_concurrent\_requests.concurrent\_program\_id%TYPE;

 v\_concurrent\_program\_name VARCHAR2(500);

 v\_log\_info\_routine\_status NUMBER;

 v\_code\_position VARCHAR2(100);

BEGIN -- Function log\_extract\_process\_info

 v\_code\_position := 'Routine Start';

 message\_output('LOG', NULL);

 message\_output('LOG', 'Logging Extract Process Information to Interface Log Table.' );

 IF gc\_debug THEN

 message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 --

 v\_code\_position := 'Query Concurrent Requests for Process Information';

 OPEN c\_concurrent\_request\_info (p\_process\_request\_id);

 FETCH c\_concurrent\_request\_info

 INTO v\_actual\_start\_date, v\_requested\_by

 , v\_responsibility\_id, v\_argument\_text

 , v\_concurrent\_program\_id, v\_concurrent\_program\_name;

 CLOSE c\_concurrent\_request\_info;

 --

 -- set completion date variable so consistent for log time and elapsed time

 -- round seconds to correct for precision rounding errors

 v\_completion\_date := SYSDATE;

 v\_code\_position := 'Convert Elapsed Time to Hours, Minutes, Seconds for Logging';

 v\_process\_elapsed\_time := elapsed\_run\_time( v\_actual\_start\_date, v\_completion\_date );

 --

 v\_code\_position := 'Convert Number Status to Word for Readability';

 IF p\_process\_control\_status = gc\_success\_status\_code THEN

 v\_request\_status := 'Normal';

 ELSIF p\_process\_control\_status = gc\_warning\_status\_code THEN

 v\_request\_status := 'Warning';

 ELSIF p\_process\_control\_status = gc\_failure\_status\_code THEN

 v\_request\_status := 'Error';

 ELSE

 v\_request\_status := NULL;

 END IF;

 --

 v\_code\_position := 'Call Routine to Log Information';

 dmhrsi.log\_interface\_information

 ( p\_request\_id => p\_process\_request\_id

 , p\_user\_concurrent\_program\_name => v\_concurrent\_program\_name

 , p\_concurrent\_program\_id => v\_concurrent\_program\_id

 , p\_process\_start\_date => v\_actual\_start\_date

 , p\_process\_completion\_date => v\_completion\_date

 , p\_requested\_by\_user\_id => v\_requested\_by

 , p\_responsibility\_id => v\_responsibility\_id

 , p\_request\_status => v\_request\_status

 , p\_argument\_text => v\_argument\_text

 , p\_process\_elapsed\_time => v\_process\_elapsed\_time

 , p\_records\_input => 0

 , p\_records\_output => p\_records\_in\_extract

 , p\_records\_processed => 0

 , p\_records\_erred => 0

 , p\_comments => NULL

 , p\_log\_information\_result => v\_log\_info\_routine\_status );

 --

 v\_code\_position := 'Post Call Routine to Log Extract Information';

 IF gc\_debug THEN

 message\_output('LOG', ' Log Interface Info Result was '|| v\_log\_info\_routine\_status ||'.');

 END IF;

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 v\_code\_position := 'Log Extract Information Completed';

 message\_output('LOG', 'Logging Extract Information to Interface Log Table Completed.' );

 --

 RETURN TRUE;

 --

 EXCEPTION

 WHEN OTHERS THEN

 message\_output('LOG', NULL);

 message\_output('LOG', ' Unexpected Error Writing to Interface Log Table.');

 message\_output('LOG', ' Process information was not added to Interface Log table.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' Error after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Exit Logging Extract Information to Interface Log Table Routine.' );

 message\_output('LOG', NULL);

 RETURN FALSE;

END log\_extract\_process\_info;

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

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

-- Function submit\_copy\_concurrent\_file

-- Submits concurrent process to copy the output file from a concurrent

-- request output file to the specified directory and filename.

-- Generates messages on errors.

--

FUNCTION submit\_copy\_concurrent\_file

 ( p\_source\_request\_id IN NUMBER

 , p\_target\_dir\_path IN VARCHAR2

 , p\_target\_file\_name IN VARCHAR2

 ) RETURN BOOLEAN IS

 --

 -- local routine variables and request variables

 --

 v\_source\_file\_name VARCHAR2(20);

 v\_cc\_request\_id NUMBER;

 v\_request\_wait\_status BOOLEAN;

 v\_request\_phase applsys.fnd\_lookup\_values.meaning%TYPE;

 v\_request\_status applsys.fnd\_lookup\_values.meaning%TYPE;

 v\_request\_dev\_phase applsys.fnd\_lookup\_values.lookup\_code%TYPE;

 v\_request\_dev\_status applsys.fnd\_lookup\_values.lookup\_code%TYPE;

 v\_request\_message VARCHAR2(500);

 v\_routine\_status BOOLEAN;

 v\_directory\_path VARCHAR2(500);

 v\_code\_position VARCHAR2(100);

BEGIN -- Function submit\_copy\_concurrent\_file

 v\_code\_position := 'Routine Start';

 message\_output('LOG', NULL);

 message\_output('LOG', 'Copying Concurrent Request Output File.' );

 v\_routine\_status := TRUE;

 IF gc\_debug THEN

 message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 --

 -- remove trailing slash off directory

 v\_directory\_path := TRIM(p\_target\_dir\_path);

 IF LENGTH(TRIM(p\_target\_dir\_path)) > 0

 AND SUBSTR(TRIM(p\_target\_dir\_path)

 ,LENGTH(TRIM(p\_target\_dir\_path))) = '/' THEN

 v\_directory\_path

 := SUBSTR(TRIM(p\_target\_dir\_path)

 ,LENGTH(TRIM(p\_target\_dir\_path))-1);

 END IF;

 --

 v\_code\_position := 'Construct Source Filename from Request ID';

 v\_source\_file\_name := 'o'|| p\_source\_request\_id ||'.out';

 message\_output('LOG', ' Copy '|| v\_source\_file\_name

 ||' to filename '|| p\_target\_file\_name

 ||' in directory '|| p\_target\_dir\_path ||'.');

 --

 -- submit request to copy concurrent request file

 v\_code\_position := 'Submit Copy Shell Script CC-Request';

 v\_cc\_request\_id

 := apps.fnd\_request.submit\_request

 ( application => gc\_dmhrsi\_appl\_short\_name

 , program => gc\_file\_copy\_cc\_prog

 , description => NULL

 , start\_time => NULL

 , sub\_request => FALSE

 , argument1 => v\_source\_file\_name

 , argument2 => gc\_concurrent\_output\_dir

 , argument3 => p\_target\_file\_name

 , argument4 => v\_directory\_path

 );

 --

 -- verify submit and respond to failure

 v\_code\_position := 'Check Return from CC-Request Submit';

 IF v\_cc\_request\_id < 0 THEN

 v\_routine\_status := FALSE;

 message\_output('LOG', NULL);

 message\_output('LOG', ' Error Encountered Submitting Request to Copy Output File.');

 message\_output('LOG', ' Program request to copy output file failed.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' - Application: '|| gc\_dmhrsi\_appl\_short\_name);

 message\_output('LOG', ' - Program: '|| gc\_file\_copy\_cc\_prog);

 message\_output('LOG', ' - Source File: '|| v\_source\_file\_name);

 message\_output('LOG', ' - Target File: '|| p\_target\_file\_name);

 message\_output('LOG', ' - Target Dir: '|| v\_directory\_path);

 ELSE

 COMMIT;

 v\_code\_position := 'Waiting on Copy File CC-Request to Finish';

 --

 -- report to user the copy file request id associated with this file

 -- and wait on request and get stats for user

 message\_output('LOG', ' The associated request for file copy of '

 || v\_source\_file\_name ||' has Request ID '|| v\_cc\_request\_id ||'.' );

 --

 v\_request\_wait\_status

 := apps.fnd\_concurrent.wait\_for\_request

 ( request\_id => v\_cc\_request\_id

 , interval => gc\_cc\_wait\_interval\_sec

 , max\_wait => gc\_cc\_max\_wait\_length\_sec

 , phase => v\_request\_phase

 , status => v\_request\_status

 , dev\_phase => v\_request\_dev\_phase

 , dev\_status => v\_request\_dev\_status

 , message => v\_request\_message

 );

 v\_code\_position := 'Checking on Copy Shell Script CC-Request Status';

 --

 -- on errors or warnings provide info to user to check log file.

 IF NOT v\_request\_wait\_status THEN

 v\_code\_position := 'Unknown Status of File Copy Request';

 message\_output('LOG', ' Could not get status of file copy.');

 message\_output('LOG', ' Please check the log file for the file'

 ||' copy and take corrective action if needed.');

 message\_output('LOG', ' Request ID: '|| v\_cc\_request\_id

 ||' for Program Name: '

 || concurrent\_request\_name(v\_cc\_request\_id) ||'.' );

 ELSE

 v\_code\_position := 'Got CC-Request Status';

 --

 -- provide feedback on status of copy

 IF v\_request\_phase = 'Completed' AND v\_request\_status = 'Normal'

 THEN

 message\_output('LOG', ' File copied Successfully.' );

 ELSE

 v\_routine\_status := FALSE;

 message\_output('LOG', ' Copy file was not successful.');

 message\_output('LOG', ' Please check the log file for the file'

 ||' copy and take corrective action if needed.');

 message\_output('LOG', ' Request ID: '|| v\_cc\_request\_id

 ||' for Program Name: '

 || concurrent\_request\_name(v\_cc\_request\_id) ||'.' );

 message\_output('LOG', ' Status is '|| v\_request\_status ||'.' );

 END IF; -- status of file copy

 END IF; -- wait function status

 END IF; -- submit status check

 v\_code\_position := 'Routine End';

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Copying Concurrent Request Output File Complete.');

 RETURN v\_routine\_status;

 EXCEPTION

 WHEN OTHERS THEN

 message\_output('LOG', NULL);

 message\_output('LOG', ' Unexpected Error Submit Request for Copying Output File.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' Error after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Exit Copying Concurrent Request Output File Routine.');

 message\_output('LOG', NULL);

 RETURN FALSE;

END submit\_copy\_concurrent\_file;

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

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

-- Procedure message\_output

-- Outputs string to log or screen based on concurrent request id test.

--

PROCEDURE message\_output

 ( p\_destination\_file IN VARCHAR2 DEFAULT 'LOG'

 , p\_message\_text IN VARCHAR2

 ) IS

 vc\_line\_length\_max CONSTANT NUMBER := 80;

BEGIN -- Procedure message\_output

 -- print to log when concurrent request

 IF NVL(fnd\_global.conc\_request\_id,0) > 0 THEN

 IF NVL(p\_destination\_file,'LOG') = 'OUTPUT' THEN

 fnd\_file.put\_line(fnd\_file.OUTPUT, p\_message\_text);

 ELSE

 fnd\_file.put\_line(fnd\_file.LOG, p\_message\_text);

 END IF;

 ELSE

 -- wrap message for non-log file

 FOR indx IN 1..CEIL(LENGTH(p\_message\_text)/vc\_line\_length\_max) LOOP

 dbms\_output.put\_line( SUBSTR(p\_message\_text, (((indx-1)\*vc\_line\_length\_max)+1), vc\_line\_length\_max) );

 END LOOP;

 END IF;

 EXCEPTION

 WHEN OTHERS THEN

 NULL;

END message\_output;

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

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

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

-- Procedure submit\_generate\_extract\_file

-- Submits concurrent process to generate the extract file.

-- Procedure returns the request id, which is needed to identify the

-- file name to copy output file.

-- Must be separate concurrent program from the main control process,

-- so the process finishes and the output file can be copied.

-- Generates message on any errors. Routine is successful unless

-- submit fails.

--

PROCEDURE submit\_generate\_extract\_file

 ( p\_period\_start\_date IN DATE

 , p\_extract\_request\_id OUT NUMBER

 , p\_routine\_status\_code OUT NUMBER

 ) IS

 --

 -- local variables

 v\_gen\_extract\_file\_status\_cd NUMBER;

 v\_code\_position VARCHAR2(100);

 v\_gen\_extract\_request\_id NUMBER;

 --

 -- submit request variables

 v\_cc\_request\_id NUMBER;

 v\_request\_wait\_status BOOLEAN;

 v\_request\_phase applsys.fnd\_lookup\_values.meaning%TYPE;

 v\_request\_status applsys.fnd\_lookup\_values.meaning%TYPE;

 v\_request\_dev\_phase applsys.fnd\_lookup\_values.lookup\_code%TYPE;

 v\_request\_dev\_status applsys.fnd\_lookup\_values.lookup\_code%TYPE;

 v\_request\_message VARCHAR2(500);

BEGIN -- Procedure submit\_generate\_extract\_file

 v\_code\_position := 'Generate Extract File Routine Start';

 message\_output('LOG', NULL);

 message\_output('LOG', 'Submit Request to Generate Extract File.');

 IF gc\_debug THEN

 message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 -- variables initialization

 v\_gen\_extract\_file\_status\_cd := gc\_success\_status\_code; -- assume success

 --

 -- submit request to generate extract file

 v\_code\_position := 'Submit Extract File Generate CC-Request';

 v\_cc\_request\_id

 := apps.fnd\_request.submit\_request

 ( application => gc\_dmhrsi\_appl\_short\_name

 , program => gc\_gen\_extract\_file\_cc\_prog

 , description => NULL

 , start\_time => NULL

 , sub\_request => FALSE

 , argument1 => p\_period\_start\_date

 );

 v\_code\_position := 'Request Id is needed for request which Copies Output File for Transfer';

 v\_gen\_extract\_request\_id := v\_cc\_request\_id;

 --

 -- verify submit and respond to failure

 v\_code\_position := 'Check Return from CC-Request Submit';

 IF v\_cc\_request\_id < 0 THEN

 v\_gen\_extract\_file\_status\_cd := gc\_failure\_status\_code;

 v\_gen\_extract\_request\_id := NULL;

 message\_output('LOG', NULL);

 message\_output('LOG', ' Error Encountered Submitting Request to Generate Extract File.');

 message\_output('LOG', ' Program request to generation of extract file failed.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' - Application: '|| gc\_dmhrsi\_appl\_short\_name);

 message\_output('LOG', ' - Program: '|| gc\_gen\_extract\_file\_cc\_prog);

 message\_output('LOG', ' - Period Start Date: '|| TO\_CHAR(p\_period\_start\_date,'DD-MON-YYYY'));

 ELSE

 COMMIT;

 v\_code\_position := 'Waiting on Generate Extract File CC-Request to Finish';

 --

 -- report to user the generate extract file request id

 -- and wait on request and get stats for user

 message\_output('LOG', ' The associated request for generate extract file'

 ||' has Request ID '|| v\_cc\_request\_id ||'.' );

 --

 v\_request\_wait\_status

 := apps.fnd\_concurrent.wait\_for\_request

 ( request\_id => v\_cc\_request\_id

 , interval => gc\_cc\_wait\_interval\_sec

 , max\_wait => gc\_cc\_max\_wait\_length\_sec

 , phase => v\_request\_phase

 , status => v\_request\_status

 , dev\_phase => v\_request\_dev\_phase

 , dev\_status => v\_request\_dev\_status

 , message => v\_request\_message

 );

 v\_code\_position := 'Checking on Generate Extract File CC-Request Status';

 --

 -- on errors or warnings provide info to user to check log file.

 IF NOT v\_request\_wait\_status THEN

 v\_code\_position := 'Unknown Status of Generate Extract File Request';

 message\_output('LOG', ' Could not get status of Generate Extract File.');

 message\_output('LOG', ' Please check the log file for the '

 ||'Generate Extract and take corrective action if needed.');

 message\_output('LOG', ' Request ID: '|| v\_cc\_request\_id

 ||' for Program Name: '

 || concurrent\_request\_name(v\_cc\_request\_id) ||'.' );

 ELSE

 v\_code\_position := 'Got CC-Request Status';

 --

 -- provide feedback on status of transfer

 IF v\_request\_phase = 'Completed' AND v\_request\_status = 'Normal'

 THEN

 message\_output('LOG', ' Generate Extract File Completed Successfully.' );

 ELSE

 v\_gen\_extract\_file\_status\_cd := gc\_failure\_status\_code;

 message\_output('LOG', ' Generate Extract File was not successful.');

 message\_output('LOG', ' Please check the log file for the'

 ||' generate extract and take corrective action if needed.');

 message\_output('LOG', ' Request ID: '|| v\_cc\_request\_id

 ||' for Program Name: '

 || concurrent\_request\_name(v\_cc\_request\_id) ||'.' );

 message\_output('LOG', ' Status is '|| v\_request\_status ||'.' );

 END IF; -- status of file generate extract

 END IF; -- wait function status

 END IF; -- submit status check

 v\_code\_position := 'Routine End';

 p\_extract\_request\_id := v\_gen\_extract\_request\_id;

 p\_routine\_status\_code := v\_gen\_extract\_file\_status\_cd;

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Submit Request to Generate Extract File Complete.');

 EXCEPTION

 WHEN OTHERS THEN

 p\_extract\_request\_id := v\_gen\_extract\_request\_id;

 p\_routine\_status\_code := gc\_failure\_status\_code;

 message\_output('LOG', NULL);

 message\_output('LOG', ' Unexpected Error Submit Request for Generate Extract File.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' Error after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 IF gc\_debug THEN

 message\_output('LOG', ' Routine End Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 END IF;

 message\_output('LOG', 'Exit Submit Request to Generate Extract File Routine.');

 message\_output('LOG', NULL);

END submit\_generate\_extract\_file;

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

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

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

-- Procedure extract\_process\_control

-- Routine to control processing of a single extract file.

-- 1. Validates Month and Year parameters

-- Month and Year to Period Start Date.

-- 2. Calls procedure to submit process to create extract file

-- 3. Calls function to submit process to copy extract file

-- Existence of Target Filename and Directory, just issues copy.

-- Must pass validation of dates for program to continue.

-- Successful submission for extract file creation and non-failure

-- completion is required for program to continue.

-- When Target infomation provided then the file copy is initiation.

-- A failure to complete the file copy will only raise the final

-- proccess status to warning.

-- Logging process information to interface log table is done during

-- extract file creation, when record counts are available.

--

PROCEDURE extract\_process\_control

 ( errbuf OUT VARCHAR2

 , retcode OUT NUMBER

 , p\_start\_date\_month IN NUMBER

 , p\_start\_date\_year IN NUMBER

 , p\_target\_file\_name IN VARCHAR2

 , p\_target\_directory IN VARCHAR2

 ) IS

 --

 -- local routine variables

 --

 v\_period\_start\_date DATE; -- start date from parameters

 v\_generate\_extract\_request\_id NUMBER;

 v\_process\_control\_status NUMBER;

 v\_routine\_start\_date DATE;

 v\_code\_position VARCHAR2(100);

 --

 -- exceptions used for parameter checks and error processing

 --

 e\_invalid\_date\_parameters EXCEPTION;

 e\_invalid\_start\_date EXCEPTION;

 e\_invalid\_future\_date EXCEPTION;

 e\_extract\_generation\_failure EXCEPTION;

 e\_debug\_halt EXCEPTION;

BEGIN -- Procedure extract\_process\_control

 v\_code\_position := 'Start CBER Extract for MDR, Proc: Process Control';

 message\_output('LOG', 'Consolidated Billing Event Repository (CBER) Extract for ');

 message\_output('LOG', 'Military Health System Data Repository (MDR).');

 message\_output('LOG', 'Request Id is '|| fnd\_global.conc\_request\_id ||'.');

 -- assume success

 v\_process\_control\_status := gc\_success\_status\_code; -- assume success

 v\_routine\_start\_date := SYSDATE;

 --

 -- when debugging which is on

 IF gc\_debug THEN

 message\_output('LOG', ' Debug mode is on.');

 message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(SYSDATE,'DD-MON-YYYY HH24:MI:SS'));

 message\_output('LOG', ' Version 18-7 Release 5.02.03.0.');

 END IF; -- process debug mode

 --

 v\_code\_position := 'Month and Year Must be Valid Before Converting to Date';

 -- verify parameters for data starting period

 IF p\_start\_date\_month IS NULL

 OR p\_start\_date\_year IS NULL

 OR p\_start\_date\_month NOT BETWEEN 1 AND 12

 OR p\_start\_date\_year > TO\_NUMBER(TO\_CHAR(SYSDATE,'YYYY'))

 THEN

 RAISE e\_invalid\_date\_parameters;

 END IF;

 v\_code\_position := 'Convert Valid Month and Year Parameters to Date';

 --

 -- have valid month and year should be valid date

 BEGIN

 v\_period\_start\_date := TO\_DATE('01'|| LPAD(p\_start\_date\_month,2,'0') || p\_start\_date\_year, 'DDMMYYYY');

 EXCEPTION

 WHEN OTHERS THEN

 RAISE e\_invalid\_start\_date;

 END;

 v\_code\_position := 'Period Date Must be Past, No Future Posting in DMHRSi';

 --

 IF v\_period\_start\_date > SYSDATE THEN

 RAISE e\_invalid\_future\_date;

 END IF;

 --

 v\_code\_position := 'Echo Parameters to Log File';

 message\_output('LOG', NULL );

 message\_output('LOG', ' Parameters');

 message\_output('LOG', ' Period Start Date used to limit records is: '

 || TO\_CHAR(v\_period\_start\_date,'DD-Mon-YYYY') ||'.' );

 --

 --

 v\_code\_position := 'To Copy Output File, Must have both Target Filename and Directory';

 -- verify parameters for target filename and directory

 IF ( ( p\_target\_file\_name IS NULL

 OR p\_target\_directory IS NULL )

 AND ( p\_target\_file\_name IS NOT NULL

 OR p\_target\_directory IS NOT NULL ) )

 THEN

 message\_output('LOG',' File Copy option requires both target filename and directory.');

 message\_output('LOG',' Target filename provided was "'|| p\_target\_file\_name ||'".');

 message\_output('LOG',' Target directory provided was "'|| p\_target\_directory ||'".');

 message\_output('LOG',' Extract file will be available in '

 ||'Concurrent Manager Output directory.');

 ELSIF p\_target\_file\_name IS NULL

 AND p\_target\_directory IS NULL

 THEN

 message\_output('LOG',' Values for optional file copy not provided.');

 message\_output('LOG',' Extract file will be available in '

 ||'Concurrent Manager Output directory.');

 ELSE

 message\_output('LOG',' Target filename and directory for copy are:');

 message\_output('LOG',' Filename: '|| p\_target\_file\_name );

 message\_output('LOG',' Directory: '|| p\_target\_directory ||'.' );

 END IF;

 --

 -- submit process to generate extract file

 -- wait for completion so output file can be copied

 v\_code\_position := 'Generate Extract file';

 submit\_generate\_extract\_file

 ( v\_period\_start\_date, v\_generate\_extract\_request\_id, v\_process\_control\_status );

 IF v\_process\_control\_status != gc\_success\_status\_code THEN

 RAISE e\_extract\_generation\_failure;

 END IF;

 --

 v\_code\_position := 'Is the Output to be Copied';

 IF p\_target\_file\_name IS NOT NULL

 AND p\_target\_directory IS NOT NULL

 THEN

 v\_code\_position := 'Have Target Parameters for File Copy';

 --

 -- copy file to requested target filename and directory

 v\_code\_position := 'Copy Extract File to Target Directory';

 IF NOT submit\_copy\_concurrent\_file

 ( v\_generate\_extract\_request\_id, p\_target\_directory, p\_target\_file\_name )

 THEN

 -- log message on failure on file copy

 message\_output('LOG', NULL);

 message\_output('LOG', ' Copy of Extract from output directory failed.');

 message\_output('LOG', ' Processing information, counts and run times '

 ||'were not logged for this request.');

 v\_process\_control\_status

 := GREATEST( v\_process\_control\_status, gc\_warning\_status\_code );

 END IF; -- submit copy

 END IF; -- test target parameters for optional file copy

 --

 v\_code\_position := 'Process Complete';

 retcode := v\_process\_control\_status;

 IF v\_process\_control\_status = gc\_success\_status\_code THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'CBER to MDR Process run elapsed time was '

 || elapsed\_run\_time( v\_routine\_start\_date, SYSDATE )

 ||' (hrs:mins:secs).');

 message\_output('LOG', 'CBER Extract for MDR Process is Complete and Successful.');

 errbuf := 'Success';

 ELSIF v\_process\_control\_status = gc\_warning\_status\_code THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'CBER Extract for MDR Process Completed with Warnings.');

 errbuf := 'Warning';

 ELSE

 message\_output('LOG', NULL);

 message\_output('LOG', 'CBER Extract for MDR Process had Exceptions or Failures.');

 errbuf := 'Failure';

 END IF;

 --

 EXCEPTION

 WHEN e\_invalid\_date\_parameters THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Month and Year parameters are required and must '

 ||'be valid values.');

 message\_output('LOG', ' Month must be between 1 and 12.');

 message\_output('LOG', ' Year cannot be greater than current year ('

 || TO\_CHAR(SYSDATE,'YYYY') ||').');

 message\_output('LOG', ' Month provided was "'|| p\_start\_date\_month ||'".');

 message\_output('LOG', ' Year provided was "'|| p\_start\_date\_year ||'".');

 errbuf := 'Invalid Date Parameters';

 retcode := gc\_failure\_status\_code;

 WHEN e\_invalid\_start\_date THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Month and Year parameters are required and must '

 ||'be valid values for a date.');

 message\_output('LOG', ' Month provided was "'|| p\_start\_date\_month ||'".');

 message\_output('LOG', ' Year provided was "'|| p\_start\_date\_year ||'".');

 errbuf := 'Invalid Date Parameters';

 retcode := gc\_failure\_status\_code;

 WHEN e\_invalid\_future\_date THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Month and Year parameters are required and must '

 ||'be a date that is current or in the past.');

 message\_output('LOG', ' The date provided was "'

 || TO\_CHAR(v\_period\_start\_date,'DD-Mon-YYYY') ||'".' );

 errbuf := 'Invalid Date Parameters';

 retcode := gc\_failure\_status\_code;

 WHEN e\_extract\_generation\_failure THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Error generation of extract file, processing stopped.');

 message\_output('LOG', ' Extract file creation failed. Please review the '

 ||'log for request id '|| v\_generate\_extract\_request\_id ||'.');

 errbuf := 'Extract Failure';

 retcode := gc\_failure\_status\_code;

 WHEN e\_debug\_halt THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Developer Halt.');

 message\_output('LOG', ' Code position '|| v\_code\_position ||'.' );

 errbuf := 'Debug';

 retcode := gc\_warning\_status\_code;

 WHEN OTHERS THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Unexpected Error in Process Control Routine.');

 message\_output('LOG', ' Please contact support and provide this log file.');

 message\_output('LOG', ' Error after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' ' || SQLERRM);

 errbuf := 'Failure';

 retcode := gc\_failure\_status\_code;

END extract\_process\_control;

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

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

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

-- Original Procedure dmhrsi.dod\_cber\_extract

-- Description from 7-Feb-2017 was:

-- Provides personnel information to MDR.

-- Only Assignments active Oct. 1, 2011 and after.

-- Pulls data active or updated during prior month

-- - Assign active (starting) on or before end of prior month

-- assgn start date < first of this month (vc\_current\_month\_start\_dt)

-- CR-16420 Title: Convert file for DMHRSi to MDR Data Feed

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

-- Procedure generate\_extract\_file

-- Routine collect personnel information in ASCII delimited file for

-- sending to MDR.

-- CR19212-Modify the feed from The Defense Medical Human Resources

-- System-Internet (DMHRSi) to the MHS Data Repository (MDR) to

-- allow for full refresh of data.

-- One input parameter is the period start date for collecting data.

-- Modify query to be full extract from a given start date (parameter).

-- Changed cursor from 4 inputs to just one. Include Assignment records

-- effective as of start date parameter. People, NPI (Qualification),

-- Person-Type, Organization-LCA and People-Group-LCA records are included

-- when they are effective during the Assignment record (effective start to

-- effective end dates).

-- Expenditure-Item-Date (effective date) used for SUOC derivation is the

-- most recent date of either the Assignment Effective Start Date or the

-- Person Duty Occupation Start Date when present.

-- At end of routine the routine run time, record counts, concurrent

-- program information is logged into the interface table. Unsuccessful

-- logging will not change the concurrent request status.

-- clear collection when person changes to keep size of collection low

-- and searches quick.

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

PROCEDURE generate\_extract\_file

 ( errbuf OUT VARCHAR2

 , retcode OUT NUMBER

 , p\_period\_start\_date IN DATE

 ) IS

 -- need distinct due to cartesian product from assignment

 -- to org lca, group lca, duty occ, person and person usage

 CURSOR c\_empl\_info\_recs

 ( cp\_period\_start\_date DATE ) IS

 SELECT DISTINCT

 pers.person\_id

 , pers.effective\_start\_date empl\_hr\_start\_date

 , pers.effective\_end\_date empl\_hr\_end\_date

 , ptuse.effective\_start\_date pers\_type\_usage\_start\_dt

 , ptuse.effective\_end\_date pers\_type\_usage\_end\_dt

 , ptuse.object\_version\_number pers\_type\_usage\_ovn

 , pers.national\_identifier

 , REPLACE(pers.attribute18,CHR(13)) edipn

 , qual\_npi.qual\_npi\_value

 , pers.employee\_number

 , pers.attribute1 person\_service

 , pers.last\_name

 , pers.first\_name

 , pers.middle\_names

 , pers.suffix

 , pers.title title\_rank

 , pers.sex gender

 , pers.attribute2 grade

 , pers.attribute4 step

 , CASE WHEN pers.attribute1 IN ('NAVY','MARINE CORPS')

 AND pers.attribute5 IS NOT NULL

 THEN 'N'||pers.attribute5

 ELSE pers.attribute5

 END person\_uic

 , pers.start\_date orig\_date\_of\_hire

 , ptyp.user\_person\_type

 , ptyp.system\_person\_type

 , asgn.assignment\_id

 , asgn.position\_id

 , asgn.job\_id

 , asgn.ass\_attribute5 skill\_type

 , asgn.ass\_attribute6 skill\_type\_suffix

 , asgn.effective\_start\_date empl\_assign\_start\_date

 , asgn.effective\_end\_date empl\_assign\_end\_date

 , asgn.organization\_id

 , CASE WHEN org\_lca.organization\_lca\_service = 'NAVY'

 AND asgn\_org.attribute13 IS NOT NULL

 THEN 'N'||asgn\_org.attribute13

 ELSE asgn\_org.attribute13

 END organization\_uic

 , org\_lca.organization\_lca\_service

 , org\_lca.organization\_lca\_dmis\_id

 , asgn.people\_group\_id

 ---, pgrp\_info.pgroup\_organization\_id

 , CASE WHEN pgrp\_lca.p\_group\_lca\_service = 'NAVY'

 AND pgrp\_info.people\_group\_uic IS NOT NULL

 THEN 'N'||pgrp\_info.people\_group\_uic

 ELSE pgrp\_info.people\_group\_uic

 END people\_group\_uic

 , pgrp\_lca.p\_group\_lca\_service

 , pgrp\_lca.p\_group\_lca\_dmis\_id

 , pgrp\_lca.p\_group\_lca\_fcc

 ---, per\_eit\_info.pers\_eit\_start\_date

 ---, per\_eit\_info.pers\_eit\_end\_date

 , per\_eit\_info.dod\_occupation\_code

 , per\_eit\_info.taxonomy\_code

 , pers.rowid person\_rowid

 , pers.creation\_date

 , pers.last\_update\_date

 , DECODE

 ( pers.attribute\_category

 , '208', '144'

 , '209', '144'

 , '210', '144'

 , '211', '143'

 , '212', '143'

 , '213', '143'

 , '214', '143'

 , '215', '123'

 , '216', '123'

 , '217', '123'

 , '218', '123'

 , '219', '142'

 , '220', '142'

 , pers.attribute\_category) primary\_pers\_type\_id

 , pers.attribute8 pers\_civilian\_flag

 , asgn.primary\_flag

 , GREATEST( asgn.effective\_start\_date

 , NVL( per\_eit\_info.pers\_eit\_start\_date, TO\_DATE('01-JAN-0001','DD-MON-YYYY') )

 ) suoc\_effective\_date

 FROM hr.per\_all\_people\_f pers

 , hr.per\_person\_types ptyp

 , hr.per\_person\_type\_usages\_f ptuse

 , hr.hr\_all\_organization\_units asgn\_org

 , hr.per\_all\_assignments\_f asgn

 LEFT OUTER JOIN

 ( -- Assigned People Group

 SELECT pgroup.people\_group\_id

 ---, pgrp\_org.organization\_id pgroup\_organization\_id

 , pgrp\_org.attribute13 people\_group\_uic

 FROM hr.pay\_people\_groups pgroup

 , hr.hr\_all\_organization\_units pgrp\_org

 WHERE pgroup.segment1 = pgrp\_org.organization\_id

 ) pgrp\_info

 ON (asgn.people\_group\_id = pgrp\_info.people\_group\_id)

 LEFT OUTER JOIN

 ( -- People Group LCA Information

 SELECT pgroup.people\_group\_id

 , pgorg\_lca.start\_date pgrp\_lca\_start\_date

 , pgorg\_lca.end\_date pgrp\_lca\_end\_date

 , DECODE( pgorg\_lca.org\_information\_context

 , 'DOD\_AF\_LCA', 'AIR FORCE'

 , 'DOD\_NAVY\_LCA', 'NAVY'

 , 'DOD\_ARMY\_LCA', 'ARMY'

 , 'DOD\_DHA\_LCA', 'DHA'

 ) p\_group\_lca\_service

 , pgorg\_lca.dmis\_id p\_group\_lca\_dmis\_id

 , pgorg\_lca.fcc p\_group\_lca\_fcc

 FROM hr.pay\_people\_groups pgroup

 , apps.dod\_lca\_classification\_v pgorg\_lca

 WHERE pgroup.segment1 = pgorg\_lca.organization\_id

 AND pgorg\_lca.end\_date >= cp\_period\_start\_date

 ) pgrp\_lca

 ON ( asgn.people\_group\_id = pgrp\_lca.people\_group\_id

 AND asgn.effective\_start\_date <= pgrp\_lca.pgrp\_lca\_end\_date

 AND asgn.effective\_end\_date >= pgrp\_lca.pgrp\_lca\_start\_date )

 LEFT OUTER JOIN

 ( -- Assigned Organization LCA Information

 SELECT olca.organization\_id

 , DECODE( olca.org\_information\_context

 , 'DOD\_AF\_LCA', 'AIR FORCE'

 , 'DOD\_NAVY\_LCA', 'NAVY'

 , 'DOD\_ARMY\_LCA', 'ARMY'

 , 'DOD\_DHA\_LCA', 'DHA'

 ) organization\_lca\_service

 , olca.start\_date org\_lca\_start\_date

 , olca.end\_date org\_lca\_end\_date

 , olca.dmis\_id organization\_lca\_dmis\_id

 FROM apps.dod\_lca\_classification\_v olca

 WHERE olca.end\_date >= cp\_period\_start\_date

 ) org\_lca

 ON ( asgn.organization\_id = org\_lca.organization\_id

 AND asgn.effective\_start\_date <= org\_lca.org\_lca\_end\_date

 AND asgn.effective\_end\_date >= org\_lca.org\_lca\_start\_date )

 LEFT OUTER JOIN

 ( -- People Extra

 SELECT ppei.person\_id

 , TO\_DATE(ppei.pei\_information1,'YYYY/MM/DD HH24:MI:SS') pers\_eit\_start\_date

 , TO\_DATE(NVL(ppei.pei\_information2,'4712/12/31 00:00:00'),'YYYY/MM/DD HH24:MI:SS') pers\_eit\_end\_date

 , jobs.attribute13 dod\_occupation\_code

 , jobs.attribute6 taxonomy\_code

 , NVL(ppei.last\_update\_date,TO\_DATE('01-JAN-0001','DD-MON-YYYY')) last\_update\_date

 FROM hr.per\_people\_extra\_info ppei

 , hr.per\_jobs jobs

 WHERE ppei.information\_type = 'DOD\_DUTY\_OCCUPATIONS'

 AND ppei.pei\_information4 = 'PRIMARY'

 AND ppei.pei\_information3 = jobs.job\_id

 AND TO\_DATE(NVL(ppei.pei\_information2,'4712/12/31 00:00:00'),'YYYY/MM/DD HH24:MI:SS')

 >= cp\_period\_start\_date

 ) per\_eit\_info

 ON ( asgn.person\_id = per\_eit\_info.person\_id

 AND asgn.effective\_start\_date <= per\_eit\_info.pers\_eit\_end\_date

 AND asgn.effective\_end\_date >= per\_eit\_info.pers\_eit\_start\_date )

 LEFT OUTER JOIN

 ( -- qualifications for npi values

 SELECT qual.attribute2 qual\_npi\_value

 , NVL(qual.start\_date,TO\_DATE('01-JAN-0001','DD-MON-YYYY')) qual\_start\_date

 , NVL(qual.end\_date,TO\_DATE('31-DEC-4712','DD-MON-YYYY')) qual\_end\_date

 , NVL(qual.last\_update\_date,TO\_DATE('01-JAN-0001','DD-MON-YYYY')) last\_update\_date

 , qual.person\_id

 FROM hr.per\_qualifications qual

 , hr.per\_qualification\_types qualtyp

 WHERE qual.qualification\_type\_id = qualtyp.qualification\_type\_id

 AND qualtyp.name = 'NATIONAL PROVIDER IDENTIFIER'

 AND qual.attribute\_category = 'NATIONAL PROVIDER IDENTIFIER'

 AND qual.attribute1 = '1' -- individual

 AND qual.attribute2 IS NOT NULL

 AND NVL(qual.end\_date,TO\_DATE('31-DEC-4712','DD-MON-YYYY'))

 >= cp\_period\_start\_date

 ) qual\_npi

 ON ( asgn.person\_id = qual\_npi.person\_id

 AND asgn.effective\_start\_date <= qual\_npi.qual\_end\_date

 AND asgn.effective\_end\_date >= qual\_npi.qual\_start\_date )

 WHERE asgn.person\_id = pers.person\_id

 --- add filters to limit by assignment range and effective by period-start-date

 --- no future dated transactions

 AND asgn.effective\_start\_date <= TRUNC(SYSDATE)

 --- records active from period-start-date

 AND asgn.effective\_end\_date >= cp\_period\_start\_date

 AND pers.effective\_end\_date >= cp\_period\_start\_date

 AND ptuse.effective\_end\_date >= cp\_period\_start\_date

 --- date-tracked records must be within range of assignment record

 AND asgn.effective\_start\_date <= pers.effective\_end\_date

 AND asgn.effective\_end\_date >= pers.effective\_start\_date

 AND asgn.effective\_start\_date <= ptuse.effective\_end\_date

 AND asgn.effective\_end\_date >= ptuse.effective\_start\_date

 AND asgn.assignment\_type = 'E'

 AND asgn.primary\_flag = 'Y'

 AND asgn.organization\_id = asgn\_org.organization\_id

 AND ptuse.person\_type\_id = ptyp.person\_type\_id

 AND pers.person\_id = ptuse.person\_id

 ORDER BY pers.person\_id

 , pers.effective\_start\_date

 , ptuse.effective\_start\_date

 , ptuse.object\_version\_number DESC

 , asgn.effective\_start\_date;

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

 /\*\*\*\*\*\*\*\*\*\*\*\*\* Local Constants \*\*\*\*\*\*\*\*\*\*\*\*/

 cv\_hr\_start\_of\_time\_date CONSTANT DATE := TO\_DATE('01-JAN-0001','DD-MON-YYYY'); -- start of time

 cv\_hr\_end\_of\_time\_date CONSTANT DATE := TO\_DATE('31-DEC-4712','DD-MON-YYYY'); -- end of time.

 cv\_ascii\_delimiter CONSTANT VARCHAR2(1) := '!';

 cv\_bulk\_collect\_limit CONSTANT NUMBER := 2000;

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

 /\*\*\*\*\*\*\*\*\*\*\*\*\* Local Variables \*\*\*\*\*\*\*\*\*\*\*\*/

 v\_gen\_extract\_file\_request\_id NUMBER;

 v\_log\_info\_routine\_status NUMBER;

 v\_generate\_extract\_status NUMBER;

 v\_routine\_start\_date DATE;

 v\_code\_position VARCHAR2(100);

 v\_output\_record\_string VARCHAR2(1000);

 v\_dummy VARCHAR2(1);

 --

 v\_extract\_record\_count NUMBER;

 v\_prev\_person\_id NUMBER;

 v\_prev\_output\_person\_id NUMBER;

 v\_prev\_ptype\_usage\_start\_dt per\_person\_type\_usages\_f.effective\_start\_date%TYPE;

 v\_prev\_ptype\_usage\_end\_dt per\_person\_type\_usages\_f.effective\_end\_date%TYPE;

 v\_prev\_ptype\_usage\_ovn per\_person\_type\_usages\_f.object\_version\_number%TYPE;

 v\_service\_unique\_occ\_code VARCHAR2(50);

 --- previous values to limited calls to suoc function

 v\_prev\_person\_service per\_all\_people\_f.attribute1%TYPE;

 v\_prev\_primary\_pers\_type\_id per\_all\_people\_f.attribute\_category%TYPE;

 v\_prev\_grade per\_all\_people\_f.attribute2%TYPE;

 v\_prev\_primary\_flag per\_all\_assignments\_f.primary\_flag%TYPE;

 v\_prev\_skill\_type per\_all\_assignments\_f.ass\_attribute5%TYPE;

 v\_prev\_skill\_type\_suffix per\_all\_assignments\_f.ass\_attribute6%TYPE;

 v\_prev\_job\_id per\_all\_assignments\_f.job\_id%TYPE;

 v\_prev\_pers\_civilian\_flag per\_all\_people\_f.attribute8%TYPE;

 v\_prev\_organization\_service apps.dod\_lca\_classification\_v.org\_information\_context%TYPE;

 v\_prev\_suoc\_effective\_date DATE;

 --

 -- table type for bulk collection

 TYPE empl\_info\_rec\_tbl

 IS TABLE OF c\_empl\_info\_recs%ROWTYPE

 INDEX BY PLS\_INTEGER;

 empl\_rec empl\_info\_rec\_tbl;

 --

 -- record and table type for associated array used to remove duplicates

 TYPE t\_unique\_output\_record\_rec IS RECORD

 ( dummy VARCHAR2(1) );

 TYPE t\_unique\_output\_record\_tbl IS TABLE OF t\_unique\_output\_record\_rec

 INDEX BY VARCHAR2(1000);

 t\_output\_record\_tbl t\_unique\_output\_record\_tbl;

 --

 -- exceptions used for checking parameter and error processing

 --

 e\_invalid\_period\_start\_date EXCEPTION;

 e\_exists\_in\_array\_is\_dup EXCEPTION;

BEGIN -- Procedure generate\_extract\_file

 v\_code\_position := 'Start CBER Extract for MDR, Proc: Process Control';

 message\_output('LOG', 'Generate CBER Extract File for MDR.');

 v\_gen\_extract\_file\_request\_id := fnd\_global.conc\_request\_id;

 message\_output('LOG', 'Request Id is '|| v\_gen\_extract\_file\_request\_id ||'.');

 -- assume success

 v\_generate\_extract\_status := gc\_success\_status\_code; -- assume success

 v\_routine\_start\_date := SYSDATE;

 --

 -- when debugging which is on

 IF gc\_debug THEN

 message\_output('LOG', ' Routine Start Time '|| TO\_CHAR(v\_routine\_start\_date,'DD-MON-YYYY HH24:MI:SS'));

 END IF; -- process debug mode

 --

 v\_code\_position := 'Verify Period Start Date is in the Past';

 IF p\_period\_start\_date > TRUNC(v\_routine\_start\_date) THEN

 RAISE e\_invalid\_period\_start\_date;

 END IF; -- process debug mode

 message\_output('LOG', ' Period Start Date used for gathering data is '

 || TO\_CHAR(p\_period\_start\_date,'DD-MON-YYYY') ||'.');

 --

 v\_code\_position := 'Initialize Variables for Counts and Previous Loop Values';

 v\_extract\_record\_count := 0;

 v\_prev\_person\_id := 0;

 v\_prev\_output\_person\_id := 0;

 v\_prev\_ptype\_usage\_start\_dt := cv\_hr\_start\_of\_time\_date;

 v\_prev\_ptype\_usage\_end\_dt := cv\_hr\_end\_of\_time\_date;

 v\_prev\_ptype\_usage\_ovn := 0;

 -- initialize prev values for suoc test

 v\_prev\_person\_service := 'XXX';

 v\_prev\_primary\_pers\_type\_id := 0;

 v\_prev\_grade := 'XXX';

 v\_prev\_primary\_flag := 'X';

 v\_prev\_skill\_type := 'XXX';

 v\_prev\_skill\_type\_suffix := 'XXX';

 v\_prev\_job\_id := 0;

 v\_prev\_pers\_civilian\_flag := 'X';

 v\_prev\_organization\_service := 'X';

 v\_prev\_suoc\_effective\_date := cv\_hr\_start\_of\_time\_date;

 --

 v\_code\_position := 'Pre-Open Cursor';

 IF c\_empl\_info\_recs%ISOPEN THEN

 CLOSE c\_empl\_info\_recs;

 END IF;

 OPEN c\_empl\_info\_recs( p\_period\_start\_date );

 -- fetch before cursor loop to keep from generating output file with zero records

 v\_code\_position := 'Pre-Fetch Cursor';

 FETCH c\_empl\_info\_recs BULK COLLECT INTO empl\_rec LIMIT cv\_bulk\_collect\_limit;

 -- do not generate output file with header only

 IF empl\_rec.COUNT = 0 THEN

 v\_code\_position := 'No Data Found Message';

 message\_output('LOG', NULL );

 message\_output('LOG', ' No records found for extract.');

 ELSE

 v\_code\_position := 'Write Header';

 -- write column header

 message\_output('OUTPUT',

 'SSN!EDIPN!NPI!Person ID!Emloyee Number!Employee Assignment DMIS ID'

 ||'!Employee People Group DMIS ID!Position ID!Job ID!Organization ID'

 ||'!People Group ID!Assignment Effective Start Date'

 ||'!Assignment Effective End Date!Person Effective Start Date'

 ||'!Person Effective End Date!Original Date of Hire'

 ||'!Person Assignment Service!People Group Service!Last Name!First Name'

 ||'!Middle Name!Suffix!Title Rank!Person Grade!Person Step!Gender!Person Type'

 ||'!Skill Type!Skill Type Suffix!Organization UIC or PAS!People Group UIC'

 ||'!DOD Occupation Code'

 ||'!Taxonomy Code 1!Taxonomy Code 2!Taxonomy Code 3'

 ||'!Taxonomy Code 4!Taxonomy Code 5!Taxonomy Code 6'

 ||'!Employee Service!Employee UIC/PAS'

 ||'!SUOC!Record ID!Creation Date!Last Update Date!People Group FCC'

 );

 END IF;

 v\_code\_position := 'Pre Cursor-Loop';

 LOOP -- process cursor

 BEGIN

 v\_code\_position := 'Top Cursor-Loop';

 -- get collection of records

 EXIT WHEN empl\_rec.COUNT = 0;

 -- process data collected

 v\_code\_position := 'Pre Collection-Loop';

 FOR indx IN 1 .. empl\_rec.COUNT LOOP

 BEGIN

 -- do not process new ex-employees

 -- do not process the person-type-usage record for retiree/beneficiary

 -- get first person-usage-type record ordered by object-version-number

 v\_code\_position := 'Collection-Loop Test Record for Output';

 IF ( empl\_rec(indx).system\_person\_type != 'EX\_EMP'

 OR v\_prev\_output\_person\_id = empl\_rec(indx).person\_id )

 AND

 ( empl\_rec(indx).person\_id != v\_prev\_person\_id

 OR empl\_rec(indx).pers\_type\_usage\_start\_dt != v\_prev\_ptype\_usage\_start\_dt

 OR empl\_rec(indx).pers\_type\_usage\_end\_dt != v\_prev\_ptype\_usage\_end\_dt

 OR empl\_rec(indx).pers\_type\_usage\_ovn = v\_prev\_ptype\_usage\_ovn )

 THEN

 v\_code\_position := 'Collection-Loop Record Will Be Output';

 v\_prev\_output\_person\_id := empl\_rec(indx).person\_id;

 -- derive suoc only if values have changed

 v\_code\_position := 'Test for Values Changed Before Deriving SUOC';

 IF v\_prev\_person\_id != empl\_rec(indx).person\_id

 OR NVL(v\_prev\_person\_service,'X') != NVL(empl\_rec(indx).person\_service,'X')

 OR v\_prev\_suoc\_effective\_date != empl\_rec(indx).suoc\_effective\_date

 OR NVL(v\_prev\_primary\_pers\_type\_id,'X') != NVL(empl\_rec(indx).primary\_pers\_type\_id,'X')

 OR NVL(v\_prev\_grade,'X') != NVL(empl\_rec(indx).grade,'X')

 OR NVL(v\_prev\_primary\_flag,'X') != NVL(empl\_rec(indx).primary\_flag,'X')

 OR NVL(v\_prev\_skill\_type,'X') != NVL(empl\_rec(indx).skill\_type,'X')

 OR NVL(v\_prev\_skill\_type\_suffix,'X') != NVL(empl\_rec(indx).skill\_type\_suffix,'X')

 OR NVL(v\_prev\_job\_id,0) != NVL(empl\_rec(indx).job\_id,0)

 OR NVL(v\_prev\_pers\_civilian\_flag,'X') != NVL(empl\_rec(indx).pers\_civilian\_flag,'X')

 OR NVL(v\_prev\_organization\_service,'X') != NVL(empl\_rec(indx).organization\_lca\_service,'X')

 THEN

 --- suoc value comes from eas-iv procedure

 v\_code\_position := 'Pre-Call for SUOC';

 v\_service\_unique\_occ\_code

 := apps.dod\_easiv\_ext.return\_suoc

 ( x\_person\_id => empl\_rec(indx).person\_id

 , x\_service\_type => empl\_rec(indx).person\_service

 , x\_expenditure\_item\_date => empl\_rec(indx).suoc\_effective\_date

 , x\_attribute\_category => empl\_rec(indx).primary\_pers\_type\_id

 , x\_grade => empl\_rec(indx).grade

 , x\_primary\_flag => empl\_rec(indx).primary\_flag

 , x\_skill\_type => empl\_rec(indx).skill\_type

 , x\_skill\_type\_suffix => empl\_rec(indx).skill\_type\_suffix

 , x\_job\_id => empl\_rec(indx).job\_id

 , x\_attribute8 => empl\_rec(indx).pers\_civilian\_flag

 , x\_charged\_service => empl\_rec(indx).organization\_lca\_service

 );

 v\_code\_position := 'Derived SUOC, Set Previous Variables';

 -- set previous values for suoc test on next loop iteration

 v\_prev\_person\_service := empl\_rec(indx).person\_service;

 v\_prev\_suoc\_effective\_date := empl\_rec(indx).suoc\_effective\_date;

 v\_prev\_primary\_pers\_type\_id := empl\_rec(indx).primary\_pers\_type\_id;

 v\_prev\_grade := empl\_rec(indx).grade;

 v\_prev\_primary\_flag := empl\_rec(indx).primary\_flag;

 v\_prev\_skill\_type := empl\_rec(indx).skill\_type;

 v\_prev\_skill\_type\_suffix := empl\_rec(indx).skill\_type\_suffix;

 v\_prev\_job\_id := empl\_rec(indx).job\_id;

 v\_prev\_pers\_civilian\_flag := empl\_rec(indx).pers\_civilian\_flag;

 v\_prev\_organization\_service := empl\_rec(indx).organization\_lca\_service;

 v\_code\_position := 'Done SUOC';

 END IF; -- value change test for suoc

 --

 --

 -- clear associated array when person changes to keep size of array low

 v\_code\_position := 'When New Person Id Clear Associative Array';

 IF v\_prev\_person\_id != empl\_rec(indx).person\_id THEN

 v\_code\_position := 'New Person Id Clear Associative Array';

 t\_output\_record\_tbl.DELETE;

 END IF; -- new person-id

 BEGIN -- block for associated array lookup for removing duplicates

 --

 -- when key information from current record exists in associative array

 -- then current record has been seen before and is a duplicate

 --

 v\_code\_position := 'Lookup Current Record Info in Array';

 -- previously seen record returns value

 -- unique record (not seen before) generates no-data-found exception

 v\_dummy

 := t\_output\_record\_tbl

 ( empl\_rec(indx).person\_id

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).user\_person\_type

 || cv\_ascii\_delimiter || empl\_rec(indx).assignment\_id

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).qual\_npi\_value

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).people\_group\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_fcc

 || cv\_ascii\_delimiter || empl\_rec(indx).dod\_occupation\_code

 || cv\_ascii\_delimiter || empl\_rec(indx).taxonomy\_code

 || cv\_ascii\_delimiter || v\_service\_unique\_occ\_code

 ).dummy;

 --

 -- record in associative array, nothing more to do with this record

 v\_code\_position := 'Seen Output Record Before - Is Duplicate';

 --

 EXCEPTION

 WHEN NO\_DATA\_FOUND THEN

 -- have unique record, add to array, increment count and output record

 v\_code\_position := 'Unique Record, Not Seen Before';

 -- this record seen and output, so add record to array

 t\_output\_record\_tbl

 ( empl\_rec(indx).person\_id

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).user\_person\_type

 || cv\_ascii\_delimiter || empl\_rec(indx).assignment\_id

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).qual\_npi\_value

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).people\_group\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_fcc

 || cv\_ascii\_delimiter || empl\_rec(indx).dod\_occupation\_code

 || cv\_ascii\_delimiter || empl\_rec(indx).taxonomy\_code

 || cv\_ascii\_delimiter || v\_service\_unique\_occ\_code

 ).dummy := 'Y';

 --

 -- increment count and output record

 v\_code\_position := 'Output Unique Record';

 v\_extract\_record\_count := v\_extract\_record\_count + 1;

 message\_output('OUTPUT', empl\_rec(indx).national\_identifier

 || cv\_ascii\_delimiter || empl\_rec(indx).edipn

 || cv\_ascii\_delimiter || empl\_rec(indx).qual\_npi\_value

 || cv\_ascii\_delimiter || empl\_rec(indx).person\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).employee\_number

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_dmis\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).position\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).job\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_id

 || cv\_ascii\_delimiter || empl\_rec(indx).people\_group\_id

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_assign\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_start\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).empl\_hr\_end\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).orig\_date\_of\_hire,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).last\_name

 || cv\_ascii\_delimiter || empl\_rec(indx).first\_name

 || cv\_ascii\_delimiter || empl\_rec(indx).middle\_names

 || cv\_ascii\_delimiter || empl\_rec(indx).suffix

 || cv\_ascii\_delimiter || empl\_rec(indx).title\_rank

 || cv\_ascii\_delimiter || empl\_rec(indx).grade

 || cv\_ascii\_delimiter || empl\_rec(indx).step

 || cv\_ascii\_delimiter || empl\_rec(indx).gender

 || cv\_ascii\_delimiter || empl\_rec(indx).user\_person\_type

 || cv\_ascii\_delimiter || empl\_rec(indx).skill\_type

 || cv\_ascii\_delimiter || empl\_rec(indx).skill\_type\_suffix

 || cv\_ascii\_delimiter || empl\_rec(indx).organization\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).people\_group\_uic

 || cv\_ascii\_delimiter || empl\_rec(indx).dod\_occupation\_code

 || cv\_ascii\_delimiter || empl\_rec(indx).taxonomy\_code

 || cv\_ascii\_delimiter || NULL -- taxonomy 2

 || cv\_ascii\_delimiter || NULL -- taxonomy 3

 || cv\_ascii\_delimiter || NULL -- taxonomy 4

 || cv\_ascii\_delimiter || NULL -- taxonomy 5

 || cv\_ascii\_delimiter || NULL -- taxonomy 6

 || cv\_ascii\_delimiter || empl\_rec(indx).person\_service

 || cv\_ascii\_delimiter || empl\_rec(indx).person\_uic

 || cv\_ascii\_delimiter || v\_service\_unique\_occ\_code

 || cv\_ascii\_delimiter || empl\_rec(indx).person\_rowid

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).creation\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || TO\_CHAR(empl\_rec(indx).last\_update\_date,'DD-MON-YYYY')

 || cv\_ascii\_delimiter || empl\_rec(indx).p\_group\_lca\_fcc

 );

 WHEN OTHERS THEN

 v\_generate\_extract\_status := gc\_failure\_status\_code;

 message\_output('LOG', ' Unexpected error in array block filtering duplicates.');

 message\_output('LOG', ' Please contact support and provide them this log file.');

 message\_output('LOG', ' Error was after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 END; -- associated array duplicates block

 --

 v\_code\_position := 'End Associative Array Block for Unique Record Control';

 --

 END IF; -- collection-loop test for new ex-employee

 v\_code\_position := 'Set Previous Varibles';

 v\_prev\_person\_id := empl\_rec(indx).person\_id;

 v\_prev\_ptype\_usage\_start\_dt := empl\_rec(indx).pers\_type\_usage\_start\_dt;

 v\_prev\_ptype\_usage\_end\_dt := empl\_rec(indx).pers\_type\_usage\_end\_dt;

 v\_prev\_ptype\_usage\_ovn := empl\_rec(indx).pers\_type\_usage\_ovn;

 v\_code\_position := 'Bottom Collection-Loop';

 EXCEPTION

 WHEN OTHERS THEN

 v\_generate\_extract\_status := gc\_failure\_status\_code;

 message\_output('LOG', ' Unexpected error in Collection Loop.');

 message\_output('LOG', ' Please contact support and provide them this log file.');

 message\_output('LOG', ' Error was after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 END;

 END LOOP; -- collection loop

 -- get data collection for next loop iteration

 v\_code\_position := 'Fetch Next Collection';

 FETCH c\_empl\_info\_recs BULK COLLECT INTO empl\_rec LIMIT cv\_bulk\_collect\_limit;

 v\_code\_position := 'Bottom Cursor-Loop';

 EXCEPTION

 WHEN OTHERS THEN

 v\_generate\_extract\_status := gc\_failure\_status\_code;

 message\_output('LOG', ' Unexpected error in Cursor Loop.');

 message\_output('LOG', ' Please contact support and provide them this log file.');

 message\_output('LOG', ' Error was after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 END;

 END LOOP; -- process cursor

 v\_code\_position := 'Post Cursor-Loop';

 CLOSE c\_empl\_info\_recs;

 --

 --

 v\_code\_position := 'Log Process Information to Log Table';

 IF NOT log\_extract\_process\_info

 ( v\_gen\_extract\_file\_request\_id

 , v\_generate\_extract\_status, v\_extract\_record\_count )

 THEN

 -- log message on failure for process information logging

 message\_output('LOG', ' Logging of extract process information failed.');

 message\_output('LOG', ' Log information for this request is not '

 ||'available in Interface Log table.');

 message\_output('LOG', ' Please reference above error and take corrective action.');

 message\_output('LOG', ' Completion Status of the Extract Request '

 ||'is not changed by the Logging failure.');

 END IF;

 --

 v\_code\_position := 'Process Complete';

 retcode := v\_generate\_extract\_status;

 IF v\_generate\_extract\_status = gc\_success\_status\_code THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Extract record count is ' || v\_extract\_record\_count ||'.');

 message\_output('LOG', 'Generate Extract run elapsed time was '

 || elapsed\_run\_time( v\_routine\_start\_date, SYSDATE )

 ||' (hrs:mins:secs).' );

 message\_output('LOG', 'Generate CBER Extract File for MDR is Complete and Successful.');

 errbuf := 'Success, Record Count '|| v\_extract\_record\_count ||'.';

 ELSIF v\_generate\_extract\_status = gc\_warning\_status\_code THEN

 message\_output('LOG', NULL);

 message\_output('LOG', 'Generate CBER Extract File for MDR is Completed with Warnings.');

 errbuf := 'Warning';

 ELSE

 message\_output('LOG', NULL);

 message\_output('LOG', 'Generate CBER Extract File for MDR had Exceptions or Failures.');

 errbuf := 'Failure';

 END IF;

 --

 EXCEPTION

 WHEN e\_invalid\_period\_start\_date THEN

 message\_output('LOG', NULL);

 message\_output('LOG', ' Invalid Period Start Date provided.');

 message\_output('LOG', ' Period Start Date cannot be in the future.');

 message\_output('LOG', ' Period Start Date provided was "'

 || TO\_CHAR(p\_period\_start\_date, 'DD-MON-YYYY') ||'".');

 retcode := gc\_failure\_status\_code;

 errbuf := 'Parameter Failure';

 WHEN OTHERS THEN

 message\_output('LOG', NULL);

 message\_output('LOG', ' Unexpected error in Routine.');

 message\_output('LOG', ' Please contact support and provide them this log file.');

 message\_output('LOG', ' Error was after code position '|| v\_code\_position ||'.' );

 message\_output('LOG', ' '|| SQLERRM );

 retcode := gc\_failure\_status\_code;

 errbuf := 'Routine Level Failure';

END generate\_extract\_file;

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

END dod\_cber\_mdr\_extract;

/