SCT Converter Tool User's Guide

SCT Application Practices

Release 3.0 / October 2003

Confidential Business Information

This documentation is proprietary information of SCT and is not to be copied, reproduced, lent or disposed of, nor used for any purpose other than that for which it is specifically provided without the written permission of SCT.

Prepared For: Release 3.0

Prepared By: SCT

4 Country View Road

Malvern, Pennsylvania 19355

United States of America

Issued: November 1999

Revised: October 2003

This publication is intended to provide accurate information regarding SCT's software. It is provided with the understanding that SCT is not engaged in rendering legal, accounting, or other professional services through the production of this publication. Further, SCT makes no claims that an institution's use of this software in accordance with this publication will insure compliance with applicable federal or state laws, rules, or regulations. SCT recommends that organizations seek professional legal advice in order to determine that their policies and practices are in compliance with applicable laws, rules, or regulations.

Because of the nature of this material, numerous hardware and software products are mentioned by name. In most, if not all, cases these product names are claimed as trademarks by the companies that manufacture the products. It is not our intent to claim these names or trademarks as our own.

Table of Contents

Chapter 1 - Introduction

Installation 1-4
DBA Issues 1-4
Brief Overview of Scripts Created by the SCT Converter Tool (CUACNVT) 1-4
Creating the Convert Table: <TargetTable>_cvt_create.sql 1-5
Creating the SQL Loader Control File: <TargetTable>_cvt.ctl 1-7
Creating the Convert Script: <TargetTable>_convert.sql 1-8

Chapter 2 - Elements of the SCT Converter Tool

Logging In 2-3
The SCT Converter Tool: CUACNVT 2-4
Parts of the CUACNVT Form 2-4
    The CUACNVT Toolbar 2-4
The Control Form 2-7
    Using Generated ID's 2-7
    Control Form File Location Feature 2-8
The Table Block 2-8
Breakpoint Window 2-10
The Column Block 2-11
Using the Convert Function Feature of CUACNVT 2-12
Applying an Oracle Function to a Column 2-12
Using a Delivered Function 2-13
The Function Code and Parameters Window 2-14
Using the Crosswalk Feature of the Converter Tool: CUACVAL 2-16
Using Options on the Crosswalk Form 2-17
Using Crosswalk Values in the Conversion Script 2-19
Loading Data into the SPRIDEN Table 2-20
Generating SCT Banner PIDM's 2-21
Using the Generated ID Feature of the Converter Tool 2-22
    Operational Concepts 2-23
Convert Script for SPRIDEN 2-23
    How It Works 2-24
    A Sample Run 2-24
Multiple Iterations of Converting SPRIDEN 2-26
Getting the PIDM for Other Tables using F_CVT_GET_PIDM 2-27
    "Use Generated ID's" not marked 2-27
    "Use Generated ID's" marked 2-28

Chapter 3 - A Conversion Example Using the SCT Converter Tool

Establishing Specifications for Conversion of SPRIDEN Table 3-5
Using CUACNVT to produce the <TargetTable>_cvt_create.sql script 3-11
Using CUACNVT to produce the <TargetTable>cvt.sql script 3-12
Using CUACNVT to produce the <TargetTable>convert.sql script 3-13
Establishing Specifications for Conversion of SPRADDR Table 3-14
Establishing Specifications for Conversion of SPBPERS Table 3-20

Chapter 4 - Scripts Produced by the SCT Converter Tool (CAUCNVT)

Create Convert Table Script: <*TargetTable*>*_cvt_create.sql* 4-3
Create SQL*Loader Script: <*TargetTable*>_cvt.ctl 4-5
Create Convert Script: <*TargetTable*>_convert.sql 4-7
Running the Convert Script 4-8
    Answering the Prompts 4-8
    Explanation of Record Type Prompts 4-8
Dealing with Errors 4-11

Appendix A - Sample Data Files Used in Examples

ASCII Data Files Used in Conversion Examples: A-3
    spriden_cvt.dat* A-3
    spraddr_cvt.dat A-3
    spbpers_cvt.dat A-4

Appendix B - Sample Table Creation Scripts

Sample <TargetTable>_cvt_create.sql scripts B-3
    spriden_cvt_create.sql B-3
    spraddr_cvt_create.sql B-4
    spbpers_cvt_create.sql B-5

Appendix C - Sample SQL*Loader Scripts

Sample <TargetTable>_cvt.ctl scripts C-3
    spriden_cvt.ctl C-3
    spraddr_cvt.ctl C-4
    spbpers_cvt.ctl C-4

Appendix D - Sample Conversion Scripts

Sample <TargetTable>_convert.sql scripts D-3
    spriden_convert.sql D-3
SPRADDR—Address Table D-17
    spraddr_convert.sql D-17
SPBPERS: General Person Biographic Demographic Table D-34
    spbpers_convert.sql D-34

Appendix E - SCT Converter Tool Functions

Viewing Functions E-3
Editing Functions Online E-4

Appendix F - Using the SCT Converter Tool

SCT Converter Tool Instructions F-3
Using CUACNVT to produce the <TargetTable>_cvt_create.sql script F-7
Using CUACNVT to produce the <TargetTable>_cvt.ctl script F-8
Using CUACNVT to produce the <TargetTable>_convert.sql script F-9

Appendix G – Utility Scripts

\util Directory Scripts G-3
Utility Script Usage G-3
SPRIDEN duplicate checking steps G-3
SPRADDR duplicate checking steps G-4
SPRTELE duplicate checking steps G-4
Extracting Data from the Audit Tables G-5
Validating Data for Crosswalked Columns G-5
Validating Data for Columns with Foreign Key Constraints G-5

Appendix H – SCT Converter Tool Functions

SCT Converter Tool Functions (alphabetically, by system) H-3

Chapter 1 - Introduction

Installation 1-4
DBA Issues 1-4
Brief Overview of Scripts Created by the SCT Converter Tool (CUACNVT) 1-4
Creating the Convert Table: <TargetTable>_cvt_create.sql 1-5
Creating the SQL Loader Control File: <TargetTable>_cvt.ctl 1-7
Creating the Convert Script: <TargetTable>_convert.sql 1-8

Introduction

In any SCT Banner conversion project, the goal is to move client data from the legacy system to the Oracle tables in the SCT Banner system.

The SCT Converter Tool is a multi-part, self-contained form (CUACNVT) that dynamically generates the basic scripts necessary to move data from legacy system flat ASCII files to Oracle tables. The conversion script produced by the SCT Converter Tool also has the capability of converting legacy system values according to user-defined specifications.

The SCT Converter Tool has many features that allow you to customize the code of the scripts it produces through entries made on the form. Some of these features include the ability to:

- Generate new SCT Banner ID numbers
- Invoke functions that will perform a variety of modifications on the legacy data
- Decode legacy values using a crosswalk form/table
- Insert user-defined constants or default values into target table columns
- Validate legacy values against system validation tables
- Change the format of incoming date values
- Check errors, and to separate and identify errored records for evaluation and correction

When you have entered all your conversion specifications through the SCT Converter Tool form (CUACNVT), the SCT Converter Tool will dynamically generate scripts to be used in the conversion process.

The SCT Converter Tool generates the 3 following scripts:

1. A script to create a convert table: <*TargetTable*>*_cvt_create.sql* The naming convention for all convert tables created is <*TargetTable*>*_cvt* where <*TargetTable*> is the actual SCT Banner table that will eventually contain the legacy data.
2. A control file used by the SQL*Loader utility to load data from the flat ASCII file into the convert table: <*TargetTable*>*_cvt.ctl* where <*TargetTable*> is the actual SCT Banner table that will eventually contain the legacy data.
3. A convert script used to move data from the convet table to the target table: <*TargetTable*>*_convert.sql* where <*TargetTable*> is the actual SCT Banner table that will eventually contain legacy data. This script also translates legacy values to appropriate target table values by applying user-specified functions, default values, and/or computed values to the target table columns during the conversion process.

The SCT Converter Tool can also produce a mapping document for each target table. This mapping document should be used to arrive at the proper structure for the flat data file.

Assumptions:

- You have a flat ASCII file from the legacy system
- Your flat file data has been mapped to an existing target table and is structured for compatibility with the target table
- If you are working through the sample exercises in this documentation, you may need to create different sample data files that are compatible with the validation codes in your training database

Installation

To install the SCT Converter Tool, please refer to the *Converter Tool Installation Guide* delivered with the SCT Converter Tool software.

DBA Issues

Before installing/using the SCT Converter Tool, you should check with your DBA for the project to discuss the issues listed below. It is possible that successful and efficient processing of the conversion script would require some adjustments by the DBA, especially when converting tables with large numbers of records. Some DBA issues:

- Creation of the appropriate user for the conversion process. A user is created specifically for the converter tool (the default user delivered with the tool installation scripts is SCTCVT). Your DBA should review the database scripts that create the user and it's privileges.
- Rollback segments to be online for processing. With the amount of data involved in a conversion, it is important for the DBA to make sure rollback segments are sized accordingly.
- Tablespace for conversion objects. The Converter Tool installation scripts create a tablespace to store tables used by the converter tool. A specific Oracle data file location and name is required prior to running the installation scripts. This is discussed further in the *Converter Tool Installation Guide*.
- Tablespace sizing. Make certain that tablespace is sized accordingly for your institution and the amount of converted legacy data.
- Buffer size (edited in the convert script, the SET SERVEROUTPUT ON command). The size of the area that stores the SQL command string. If an ORU-10027 Buffer Overflow error occurs when running a conversion script, change the SIZE option in the SETSERVEROUTPUT ON command.

  Example:

  SET SERVEROUTPUT ON SIZE 100000

The CREATE TABLE command will include changes in a future release of the SCT Converter Tool.

Brief Overview of Scripts Created by the SCT Converter Tool (CUACNVT)

This section offers a brief overview of the scripts produced by the SCT Converter Tool. Details about the scripts appear in the section entitled "Scripts Produced by the SCT Converter Tool". Full text of the scripts from the sample conversion appears in Appendix B: Sample Table Creation Scripts, Appendix C: Sample SQL*Loader Scripts, and Appendix D: Sample Conversion Scripts.

Creating the Convert Table: *<TargetTable>_cvt_create.sql*

The first script you create using the SCT Converter Tool and execute in SQL*Plus creates an Oracle convert table to hold the data from the client's flat file and converted (transformed) values that can be loaded directly into the SCT Banner target table.

The CUACNVT form allows you to specify columns that will be loaded during the conversion and to define transformation rules to be applied during the conversion process. These should include all columns for which you have legacy data, as well as columns for which there will be default values and/or functions.

After you have defined your conversion rules, you can then produce a script (*<TargetTable>_cvt_create.sql)* that creates a convert table into which the legacy data will be loaded.

The table creation script contains each column in the target table, as well as a *convert* column for each of the columns in the target table. The *convert* columns for the script are created by the Converter Tool. They are exact replicas of the target table columns in size and position but without the constraints that exist on the target table columns. These values can be changed by the user based on the field size and data element order of the data in the ASCII flat file from the legacy system. *Convert* columns are always the VARCHAR2 datatype. If the actual SCT Banner column is not a character data type, the Converter Tool will create a VARCHAR2 column equivalent to the length of the SCT Banner column. For example, a SCT Banner date column would have a convert column created that is nine characters long because the default Oracle date format ("DD-MON-YY") is nine characters long. It is essential to adjust the size for each *convert* column based on the size of the data in the ASCII flat legacy data files.

This is a sample of a convert table creation script, *spriden_cvt_create.sql,* generated from the CUACNVT form.

```
/*
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   08/10/2002
  This script creates the conversion table SPRIDEN for
  converting legacy data into SPRIDEN.  This script was generated
  by the SCT Converter Tool.

AUDIT TRAIL END
*/
DROP TABLE SPRIDEN_CVT;
CREATE TABLE SPRIDEN_CVT
(
SPRIDEN_PIDM NUMBER(8),
CONVERT_PIDM VARCHAR2(8),
SPRIDEN_ID VARCHAR2(9),
CONVERT_ID VARCHAR2(9),
SPRIDEN_LAST_NAME VARCHAR2(60),
CONVERT_LAST_NAME VARCHAR2(60),
SPRIDEN_FIRST_NAME VARCHAR2(15),
CONVERT_FIRST_NAME VARCHAR2(15),
SPRIDEN_MI VARCHAR2(15),
CONVERT_MI VARCHAR2(15),
SPRIDEN_CHANGE_IND VARCHAR2(1),
CONVERT_CHANGE_IND VARCHAR2(1),
SPRIDEN_ENTITY_IND VARCHAR2(1),
CONVERT_ENTITY_IND VARCHAR2(1),
SPRIDEN_ACTIVITY_DATE DATE,
CONVERT_ACTIVITY_DATE VARCHAR2(9),
SPRIDEN_USER VARCHAR2(30),
```

```
CONVERT_USER VARCHAR2(30),
SPRIDEN_ORIGIN VARCHAR2(30),
CONVERT_ORIGIN VARCHAR2(30),
SPRIDEN_SEARCH_LAST_NAME VARCHAR2(60),
CONVERT_SEARCH_LAST_NAME VARCHAR2(60),
SPRIDEN_SEARCH_FIRST_NAME VARCHAR2(15),
CONVERT_SEARCH_FIRST_NAME VARCHAR2(15),
SPRIDEN_SEARCH_MI VARCHAR2(15),
CONVERT_SEARCH_MI VARCHAR2(15),
SPRIDEN_SOUNDEX_LAST_NAME CHAR(4),
CONVERT_SOUNDEX_LAST_NAME VARCHAR2(4),
SPRIDEN_SOUNDEX_FIRST_NAME CHAR(4),
CONVERT_SOUNDEX_FIRST_NAME VARCHAR2(4),
SPRIDEN_NTYP_CODE VARCHAR2(4),
CONVERT_NTYP_CODE VARCHAR2(4),
SPRIDEN_CVT_RECORD_ID NUMBER(8) CONSTRAINT PK_SPRIDEN_CVT
PRIMARY KEY,
SPRIDEN_CVT_STATUS VARCHAR2(1),
SPRIDEN_CVT_JOB_ID NUMBER(8) )
STORAGE (INITIAL 1M
      PCTINCREASE 100
      MAXEXTENTS 50);
```

Notice the *CONVERT_* column after each SPRIDEN column, using the naming convention that substitutes the word *"CONVERT"* for the SPRIDEN table name.

The legacy data from an ASCII flat file is loaded into the *convert* columns in the convert table using SQL*Loader and the SQL*Loader control file produced from CUACNVT.

The "Length" field allows you to modify the length of the convert columns to accommodate longer legacy values than would be allowed in the target table. The data can then be manipulated within the convert table as necessary using a variety of functions to be made compatible with the structure and constraints of the target table.

Note: If the legacy data files are fixed position or fixed width, the length specified in the CUACNVT form must be exactly the same length as the field in the incoming legacy data file. If the legacy data files are variable width and delimited by a special character (e.g. a comma) the Length field must be at least as long as the longest known legacy value for that column but can be longer.

Notice also that 3 additional columns, not part of the original table, appear in the convert table:

- The <*TargetTable*>*_cvt_record_id* column holds a sequence number for each record which is assigned when the table is loaded using the SQL*Loader control file produced by the SCT Converter Tool.
- The <*TargetTable*>*_cvt_status* column holds a code that indicates the conversion status of the record. In the initial loading of the legacy data to the convert table, the status is set to "N" for (N)ew. During the conversion process, the status can change to (E)rrored or (C)onverted. Finally, once the data has been inserted into the target table, the value in this column becomes "I" for (I)nserterd. These values and their use are explained in more detail in the section called "Scripts Produced by the SCT Converter Tool."
- The <*TargetTable*>*_cvt_job_id* column holds a unique number which is used in naming the log files created for each run of the convert script, allowing the log files not to be overwritten each time the convert process is run.

The <*TargetTable*>_cvt_create.sql script should be run in SQL on the server in the correct database instance to create the convert table.

Creating the SQL Loader Control File: *<TargetTable>_cvt.ctl*

A second script created by the converter tool is the loader script, or control file, that is used by the SQL*Loader utility to load the legacy data into the *convert* columns in the *<TargetTable>_cvt* table.

To indicate which columns should be included in the control file, the user can check the "Load" indicator on the CUACNVT form. The resulting control file will contain a listing of the *convert* columns that parallel the target table columns selected on the CUACNVT form, and marked with the "Load" indicator. Also included are the *<TargetTable>_convert_record_id* and the *<TargetTable>_cvt_status* columns.

NOTE: Check the "Load" indicator only for columns for which there is legacy data. If there is no legacy data for a target table column, or if the column will be populated using a database function, do not check the "Load" indicator. In our sample conversion, this is the flat file that contains the data to be loaded into the *spriden_cvt* convert table by the loader process:

spriden_cvt.dat
Legacy Fields: legacy ID, legacy ID, last name, first name, middle name
SCT Banner Fields: spriden_id, spriden_id, spriden_last_name, spriden_first_name, spriden_mi, spriden_change_ind, spriden_entity_ind

```
DATA:.....
,111111111,SMITH,AMELIA,KAY,,P
111111111,101101101,SMITH,AMELIA,KAY,I,P
          ,222222222,SIMONEAUX,MARYANNE,FRANCIS,,P
222222222,222222222,CAMPESI,MARYANNE,S,N,P
          ,333333333,WOODRUFF,LUCILLE,MARIE,,P
          ,444444444,CAMPBELL,JOSEPH,ALEXANDER,,P
          ,555555555,PITTMANN,MILDRED,YATES,,P
          ,666666666,BOYD,WILLIAM,GUERRY,,P
          ,777777777,GLOVER,SAVION,TAP,,P
          ,888888888,WINGFIELD,WILLIAM,DOUGLASS,,P
          ,999999999,LENNON,JOHN,O,,P
          ,000000000,ELDRIDGE,GROVER,BLUE,,P
```

Note: Please refer to the "Loading Data into the SPRIDEN Table" section of this document for a detailed explanation of the file layout for the spriden table.

This is a sample control file, *spriden_cvt.ctl,* generated by the converter tool for the SPRIDEN table after selecting the desired columns on the CUACNVT form by checking the "Load" indicator:

```
-- AUDIT TRAIL
-- Converter Tool:  2.10                SCTCVT    09/30/2002
--    This SQLLoader control file is used to load legacy data into
--    the conversion table SPRIDEN_cvt.  This script
--    was generated by the SCT Converter Tool.
--
-- AUDIT TRAIL END
Load data
Infile 'spriden_cvt.dat'
APPEND
INTO TABLE SPRIDEN_CVT
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' TRAILING
NULLCOLS
(
CONVERT_PIDM CHAR,
```

```
CONVERT_ID CHAR,
CONVERT_LAST_NAME CHAR,
CONVERT_FIRST_NAME CHAR,
CONVERT_MI CHAR,
CONVERT_CHANGE_IND CHAR,
SPRIDEN_CVT_RECORD_ID SEQUENCE(MAX,1),
SPRIDEN_CVT_STATUS CONSTANT 'N'
)
```

The control file includes *convert* columns for the legacy data fields from the flat file in the correct order, as specified on the CUACNVT form with the "Load" indicator.

The control file and the data source file should be moved to the same directory on the database server. SQL*Loader should be run on the server from the directory where both files reside. For detailed instructions on using SQL*Loader to load the convert tables, refer to Appendix F: Instructions for Using the SCT Converter Tool. Additionally, refer to the Oracle *Server Utilities* documentation for more information about using SQL*Loader.

Note: Disable Archive Log Mode (when using SQL*Loader to load data to a production database). Archive Log Mode impedes performance of large data loads.

Creating the Convert Script: *<TargetTable>_convert.sql*

A third script produced by the SCT Converter Tool is the conversion script. This script is run on the server after the legacy data from the flat file has been loaded into the *convert* columns of the convert table. The convert script can be run in three modes:

"Convert" mode: copies data from *convert* columns to <*TargetTable*> columns in convert table and transforms according to transformation rules set up by user;

"Insert" mode: inserts converted data into <*TargetTable*>, columns in actual target table;

"Both" mode: converts and inserts data into the target table in the same transaction.

The convert script, <*TargetTable*>*_convert.sql,* incorporates all the specifications made in the CUACNVT form for columns marked with the "Insert" indicator, such as functions, default values for columns, decodes/crosswalk values, etc.

When the <*TargetTable*>*_convert.sql* script is run on the server in "Convert" mode, it copies the values from the *convert* columns in the convert table to the <*TargetTable*> columns in the convert table and applies all functions, default values, etc. to the <*TargetTable*> columns. You can then observe any errors that occur when the legacy data is moved to the <*TargetTable*> columns in the convert table where the constraints *are* enabled.

Records that cannot be successfully moved to the <*TargetTable*> columns in the convert table are marked with an error indicator ("E") in the <*TargetTable*>*_cvt_status* column and displayed in the "Conversion Errors" window (CUICERR) of the CUACNVT form. Error feedback is also given online after the convert script has run. Users can also view the errors in the log file or from a sql session, selecting from the error table *curcerr*.

The intermediate stage of populating the <*TargetTable*> columns in the convert table allows you to evaluate the results of the conversion without working in the actual target tables, correct the errors, and re-run the conversion script until all records are successfully loaded into the <*TargetTable*> columns of the convert table.

Once all errors have been corrected or handled in some other way, the convert script is run in "Insert" mode. This process inserts the data from the convert table <*target_table*> columns into the actual target table.

There is the option for the convert script to both “Convert” and “Insert” in the same transaction. This is convenient for loading small, simple tables such as validation tables, where there is a strong confidence in the accuracy and compatibility of the data with the target table. For most conversions, however, it is recommended that the script be run in “Convert” mode first to determine if errors exist.

For a detailed look at the convert script, refer to the “Scripts Produced by the SCT Converter Tool” section of this document, as well as Appendix D: Sample Conversion Scripts.

Chapter 2 - Elements of the SCT Converter Tool

Logging In 2-3
The SCT Converter Tool: CUACNVT 2-4
Parts of the CUACNVT Form 2-4
The CUACNVT Toolbar 2-4
The Control Form 2-7
Using Generated ID's 2-7
Control Form File Location Feature 2-8
The Table Block 2-8
Breakpoint Window 2-10
The Column Block 2-11
Using the Convert Function Feature of CUACNVT 2-12
Applying an Oracle Function to a Column 2-12
Using a Delivered Function 2-13
The Function Code and Parameters Window 2-14
Using the Crosswalk Feature of the Converter Tool: CUACVAL 2-16
Using Options on the Crosswalk Form 2-17
Using Crosswalk Values in the Conversion Script 2-19
Loading Data into the SPRIDEN Table 2-20
Generating SCT Banner PIDM's 2-21
Using the Generated ID Feature of the Converter Tool 2-22
Operational Concepts 2-23
Convert Script for SPRIDEN 2-23
How It Works 2-24
A Sample Run 2-24
Multiple Iterations of Converting SPRIDEN 2-26
Getting the PIDM for Other Tables using F_CVT_GET_PIDM 2-27
"Use Generated ID's" not marked 2-27
"Use Generated ID's" marked 2-28

Elements of the SCT Converter Tool

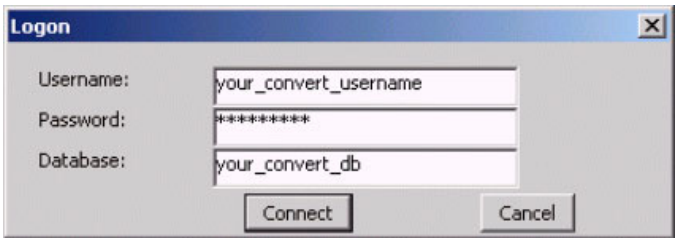
Logging In

If you are using the client-server version of the tool, double-click the SCT Converter Tool icon that was created on your PC's desktop during installation.

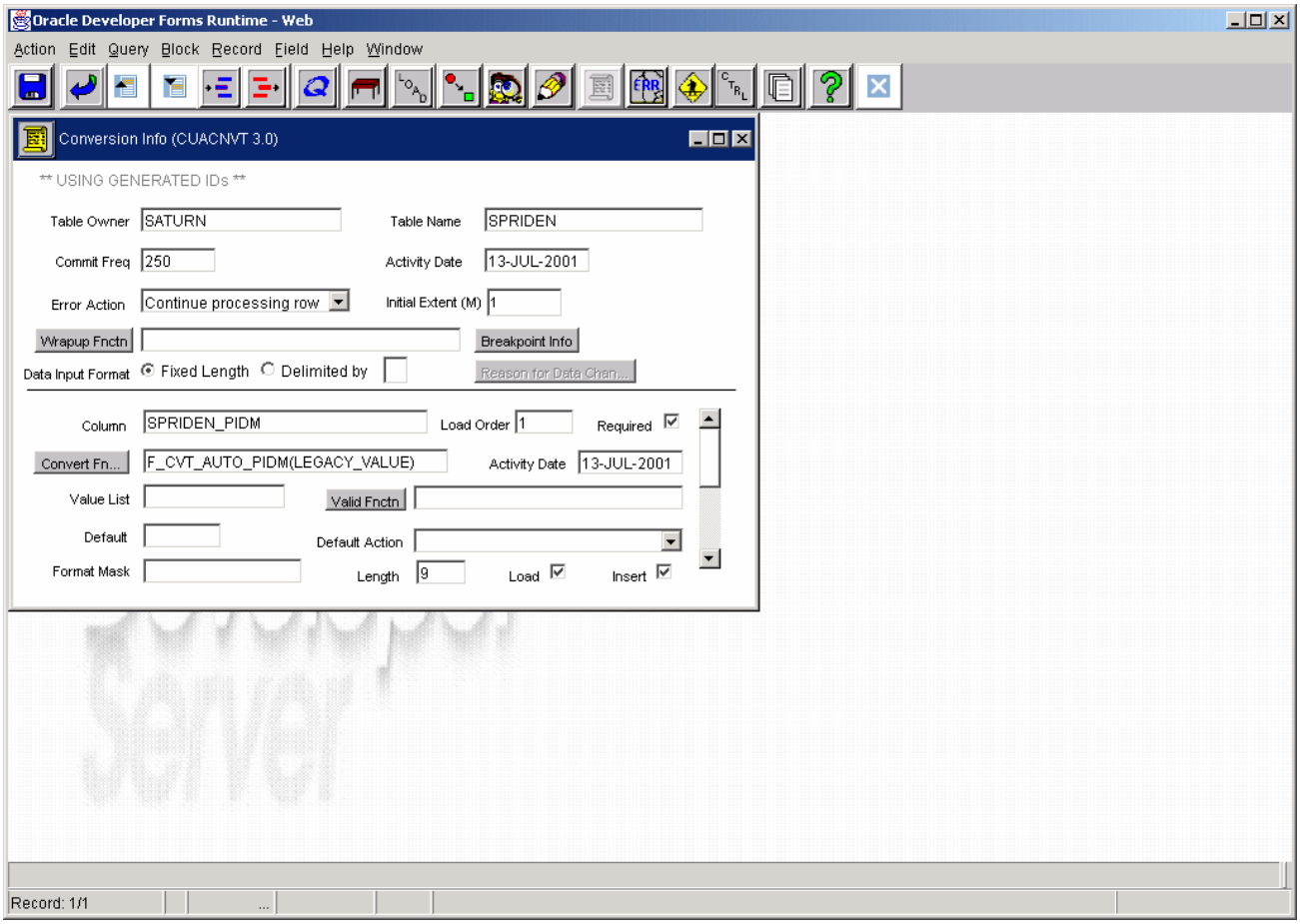
If you are using the Internet Native (IN) version of the tool, launch Internet Explorer and access the site where the IN version of the tool reside by typing it on the address window of the Internet Explorer.

Log in.

Username:	your_convert_username (provided by your DBA)
Password:	your_convert_user password (provided by your DBA)
Database:	your_convert_db (desired database, using correct alias in tnsnames.ora file)



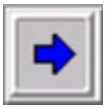
The SCT Converter Tool: CUACNVT



Parts of the CUACNVT Form

The CUACNVT Toolbar

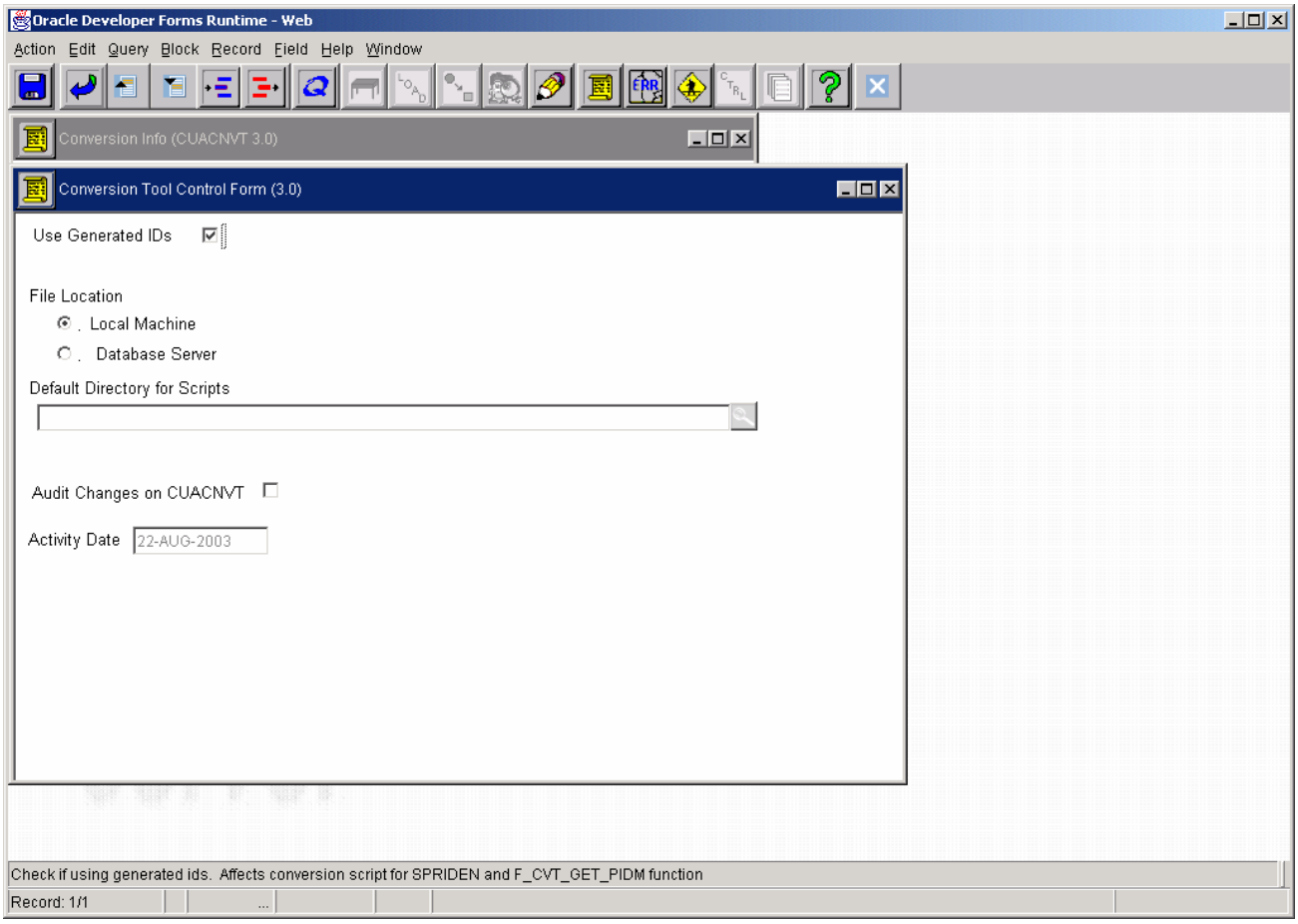
Icon	Description
 **Save**	Saves all changes entered since the last save.
 **Clear Form**	Clears all information from form so that query or new record can be entered.

Icon	Description
 **Previous Record**	Moves cursor to the first enterable field in the previous record.
 **Next Record**	Moves cursor to the first enterable field in the next record of the current information area.
 **Delete Record**	Deletes the current record.
 **Insert Record**	Inserts a new blank record.
 **Query**	Allows you to enter and execute queries.
 **Convert Table Script**	Creates <*TargetTable*>*_cvt_create.sql* script that creates an Oracle convert table according to user-entered specifications.
 **SQL*Loader Script**	Creates <*TargetTable*>*_cvt.ctl* script that loads legacy data from client flat ASCII file to convert table using SQL*Loader.
 **Conversion Script**	Creates <*TargetTable*>*_convert.sql* script that transforms and moves legacy data from convert table to target table.
 **View Function**	Allows you to view function text, to enter function parameters, and to construct function call to be included in <*TargetTable*>*_convert.sql* script.

Icon	Description
 **Edit Function**	Allows you to edit a function.
 **Converter Tool Main Window**	Returns user to the CUACNVT form from other windows in the Converter Tool.
 **View Errors**	Shows records that were not converted during conversion processing.
 **View/Edit Crosswalk**	Allows you to set up a crosswalk of values to be decoded from legacy to valid target table values.
 **Control Form**	Used to access the Converter Tool Control form, where a variety of default session values can be set up.
 **Generate Report**	Generates a report that displays the contents of the table and column block rules. This document can be used to map legacy data to target table columns or can be used as conversion archive report.
 **Help**	Displays the Help window, which provides Overall help. Clicking Help on the Menu brings up context sensitive help.
 **Exit Form**	Closes CUACNVT and logs user off.

The Control Form

The Conversion Tool Control Form is the first form that the user should access when setting up rules in the converter tool. It is accessed by the CTRL button on the tool bar. The Control Form allows users to enter several global settings that will affect the entire conversion. From the control form, users can specify whether or not to use SCT Banner generated ID numbers and can specify a default directory for all Converter Tool output, either on the local machine or on the database server.



Using Generated ID's

Check the "Use Generated IDs" box if you wish to generate new SCT Banner IDs as records are loaded during the conversion. For a detailed explanation of generating IDs during a conversion, please see the "Using the Generated ID Feature of the Converter Tool" section of this document.

Control Form File Location Feature

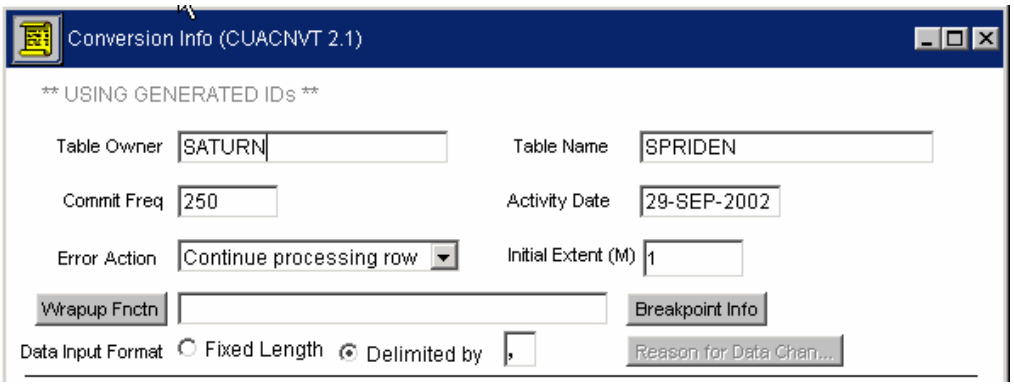
Users are able to specify a default directory for Converter Tool output files on the Control Form.

Local Machine: In client/server mode, the local machine will be the hard drive of the user's local computer. In Internet Native mode, the local machine will be the forms server where the Converter Tool client objects are located. Since this may create some difficulties with access in some situations, it is recommended that users write to the database server by choosing the "Database Server" option.

Database Server: This option uses the Oracle UTL_FILE package, which reads and writes files to the database server. In order to use this option, the user must have an entry in the init.ora file. By choosing "Database Server, " the scripts are written directly to the database server, thus eliminating the need to FTP files from the user's computer to the database server.

The Table Block

The top block of CUACNVT is the Table Block. To begin using the form, choose the target table into which you will be inserting legacy data.



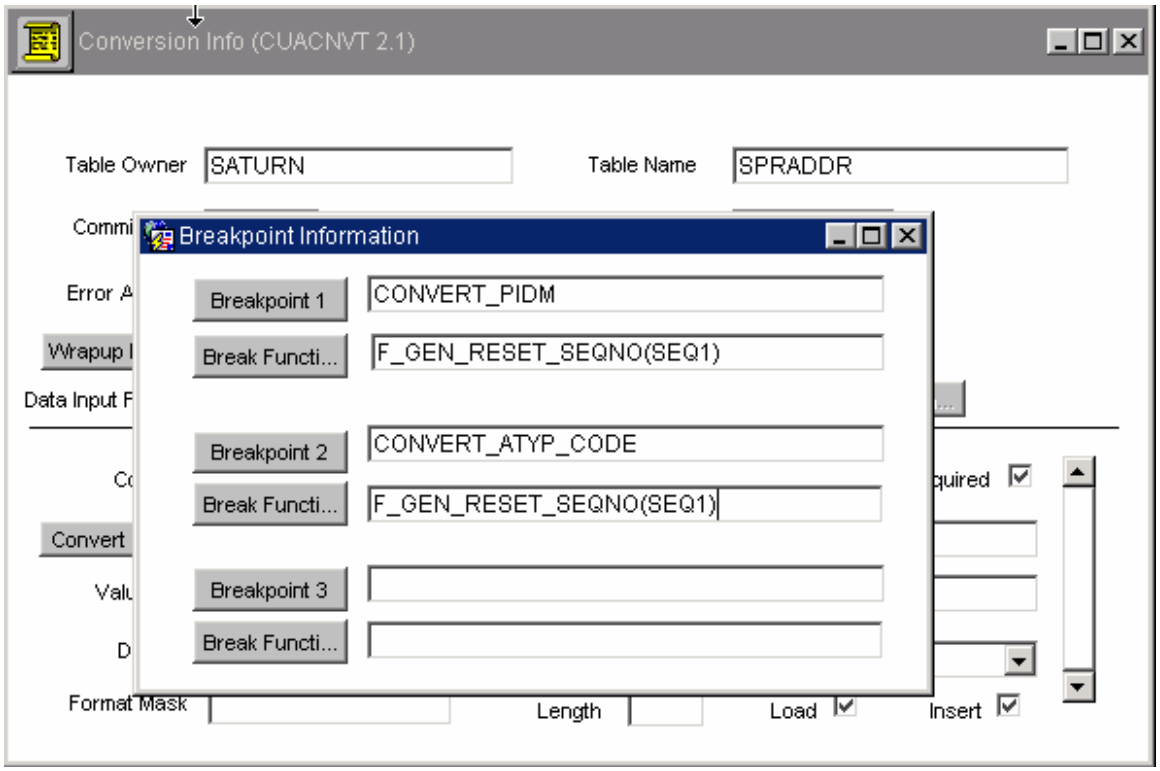
Fields	Description
Table Owner	Enter the name of the owner of the target table (SATURN, FIMSMGR, TAISMGR, etc.).
Table Name	Enter the name of the target table.
Commit Freq	Enter the number of rows for the commit interval for the convert script. A commit will be performed every "*x*" rows during the run of the <*target_table_convert.sql*> script.
Activity Date	The system date defaults here.
Error Action	The two options are: Continue processing row (default) Terminate processing of row
Wrap-up Function	Allows you to insert a function into the conversion script that will accomplish additional processing after all rows are processed. For example, in Academic History conversions, it would be possible to insert a function that would calculate g.p.a. and insert it into the proper column.

Fields	Description
Breakpoint Info	Commonly used with sequences. The window has two parameters that must be entered: Breakpoint(s) --the column(s) on which the break is to occur for setting a breakpoint function. For example, if you wish to apply sequence numbers within each occurrence of pidm and term code, and break (reset the sequence number) on a change in pidm and term code, your breakpoints would be convert_pidm and convert_term_code. Breakpoints should be set on the CONVERT columns rather than the target-table columns. Break Function– the function that will re-set the sequence based on a change in value in the breakpoint(s) established in the previous parameter. **Note:** There are 3 sequences delivered with the Converter Tool server side objects. When using breakpoints to reset sequences, be sure to use the same sequence number that was used to set the initial sequence.
Data Input Format	Specify here the format of the legacy data file. Options are "Fixed Length" or "Delimited." "Delimited" is the default value. If data is delimited, users must enter a value in the "Delimiter" field before proceeding.
Reason for Data Change	Available in future version.
Initial Extent	Allows users to set the initial extent for the convert table.

After entering all necessary data in the Table Block, SAVE the changes. Perform NEXT BLOCK function to move the to Column Block.

The column block is automatically populated with all columns from the target table specified in the Table Block.

Breakpoint Window



Note: The breakpoint function uses the column(s) in each Breakpoint field to order the conversion. Each time the breakpoint value(s) changes, the function is executed. Use CONVERT columns (CONVERT_PIDM, CONVERT_ATYP_CODE) to indicate the breakpoints. Use the F_GEN_RESET_SEQNO function to reset the sequence numbers. The sequence parameter passed should match the sequence number passed as a parameter to the F_GEN_SEQNO function on the column where the sequence number is being set. For example, to set a sequence number for address types on SPRADDR_SEQNO, use F_GEN_SEQNO(SEQ1), and set the breakpoints as CONVERT_PIDM, CONVERT_ATYP_CODE using F_GEN_RESET_SEQNO(SEQ1), as illustrated.

Breakpoints 1, 2, and 3 represent the column(s) on which the break in sequence or calculation should occur.

Break Function represents the function that re-sets the sequence or performs the calculation.

Users have the option of breaking on up to 3 key fields.

The Column Block

Fields	Description
Column	This field contains each column in the target table specified in the Table Block
Load Order	This is a sequence number that denotes the position of the column in the legacy data file. It can be changed if necessary to reflect the order of the data in the data file. A change in this field will affect the order of the columns in the %_cvt table and in the control file.
Required	This indicates whether or not the column is a "NOT NULL" column in the target table.
Convert Fnctn	This field allows you to insert function calls in the conversion script that will manipulate the data in that column. You may insert any Oracle function, such as "UPPER," "INITCAP," "TO_DATE," etc., may use any of the delivered functions, or may write new functions. For detailed instructions on using the "Convert Function" feature of CUACNVT, please refer to the "Using the Convert Function Feature" section of this document.
Activity Date	This field indicates the date the conversion rule record was created or updated. This is NOT the activity date column in the actual SCT Banner target table. Activity date for the target table is populated separately through loading of the *<TargetTable>_activity_date* field.
Value List	Allows you to specify a list of values against which legacy values can be validated. Correct format is: "value1", "value2", "value3",… In the convert script, this list follows the IN operator. If a value in the legacy row does not appear in the list, an error will occur in the row, but processing will continue if you have chosen "Continue Processing Row" as your "Error Action" in the Table Block of the form. Data entered into this field becomes part of a standard Oracle IN statements. NOTE: Do not enclose data in parentheses.
Validate Function	Allows you to choose a function that will validate all incoming legacy values against a target validation table. Use the F_VALIDATE_SINGLE function from the function list.
Default	Allows you to enter a default value for a column. Values should not be enclosed in quotes.
Default Action	Required if default value is entered. Choices are: **(D)**efault if no legacy passed **(S)**ubstitute if no value/error **(O)**verride any legacy passed

Fields	Description
Format Mask	Enter format of legacy date value so that it will be recognized and appropriately translated as a date. Example: Legacy date value format = ddmmyy Enter this format mask in "Format Mask" field. Convert script will recognize it as a date and translate the value to proper system date format.
Length	Allows you to set the length of the *convert* columns in the convert table, if they need to be larger than the target table columns in order to receive legacy data for manipulation.
Load	Allows you to indicate that you wish the current column to appear in the SQL*Loader script. The columns marked with the "load" indicator should be only those that have values in the legacy data flat file.
Insert	Allows you to indicate that you wish the current column to appear in the Conversion Script to be inserted into the target table. The columns marked with the "insert" indicator should be all those which will be inserted into the *convert* columns and subsequently into the *target table* columns. An *inserted* column will not necessarily be a *loaded* column. *inserted* values could come from defaults or from functions as well as from legacy data.

Using the Convert Function Feature of CUACNVT

The "Convert Fnctn" field allows you to insert function calls in the convert script that will manipulate the data in a selected column during the conversion process.

You may insert any baseline Oracle function, such as "UPPER," "INITCAP," "TO_DATE," etc., use any of the functions delivered with the converter tool, or you may write new functions.

To use new or existing functions not delivered with the Converter Tool, you need only to compile the function(s) in the database where you are working under the owner you are using to do the conversion Your function name should appear in the list of functions the next time you log in to the Converter Tool.

Applying an Oracle Function to a Column

Insert the function and its parameters directly into the "Convert Fnctn" field.

For example, if the legacy data is in all UPPER case, you may wish to apply the INITCAP function to the SPRIDEN_LAST_NAME field. Enter the following value into the "Convert Fnctn" field:

```
INITCAP(LEGACY_VALUE)
```

This will apply Oracle's INITCAP function to the legacy value in the *convert_last_name* field of the *spriden_cvt* table during the conversion process, causing the value inserted into the *spriden_last_name* field of the convert table and subsequently the *spriden_last_name* field in the actual SPRIDEN table to be mixed case.

Using a Delivered Function

With the cursor in the "Convert Fnctn" field, click the CONVERT FNCTN button and choose a function name from the list. For this example, we will use the F_CVT_GET_PIDM function. It is returned to the field.

The F_CVT_GET_PIDM function is used to populate the pidm of the target table. This function looks at the legacy ID, which resides in the *convert_pidm* column of the convert table. The function looks for a match in the SPRIDEN table in the SPRIDEN_ID column, then returns the SPRIDEN_PIDM to be used to populate the <*TargetTable*>*_pidm* column of the convert table. If a match on ID is not found, an error message is generated. The assumption is that the SPRIDEN table will be loaded first so that SPRIDEN_ID can be used to fetch pidms. Please note that the function will return a match to a legacy id (where the change indicator is set to "I" and the spriden_ntyp_code is set to "LGCY") if the "Generate IDs" option is chosen on the CUACTRL control form when setting up rules for converting the SPRIDEN table.

The length of the *convert_pidm* column must be adjusted to accommodate legacy IDs longer than 8 characters. To adjust the length of any column, go to the "Length" field in the column block of CUACNVT and change the number. All convert columns are VARCHAR2 datatypes.

For more information on the structure of the convert table and an explanation of the *convert* columns, please see "Scripts Produced by the SCT Converter Tool."

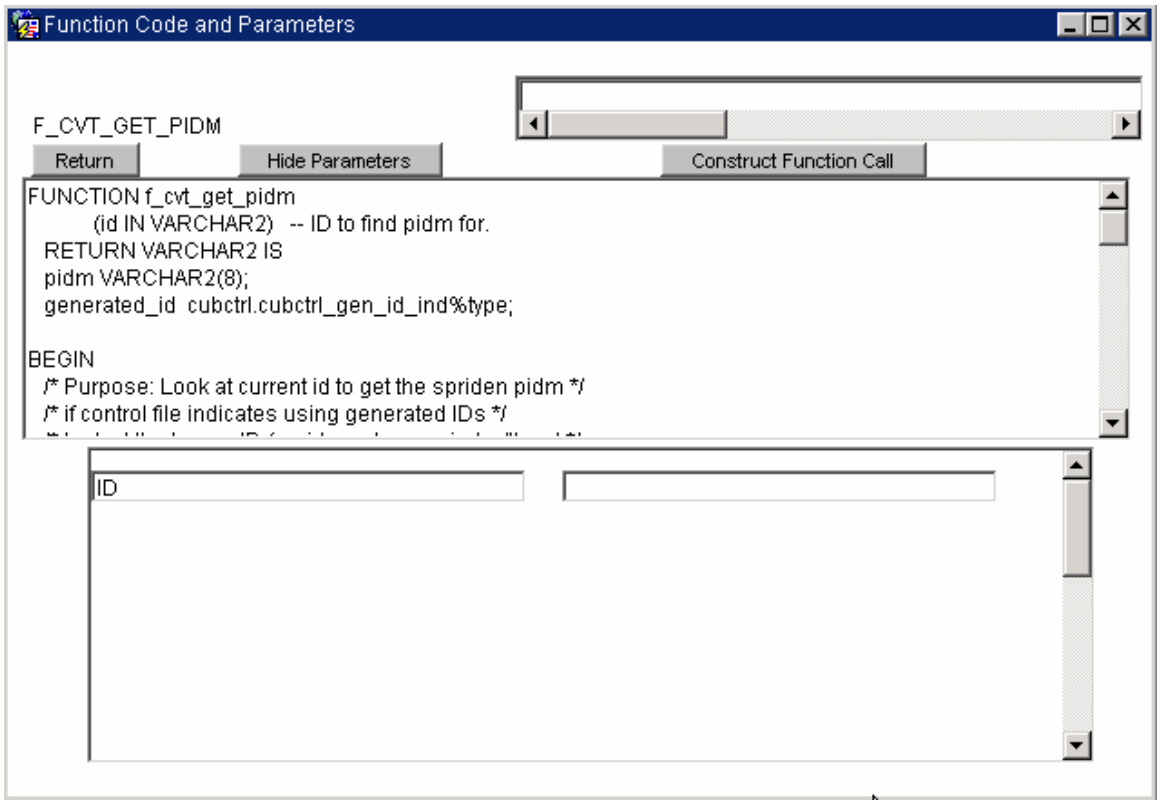
To continue, click the VIEW FUNCTION button on the Tool Bar (response is not immediate—please wait a few seconds). The "Function Code and Parameters" window appears.

The Function Code and Parameters Window

This window allows you to view the text of the function, enter parameters, and constructthe function call.

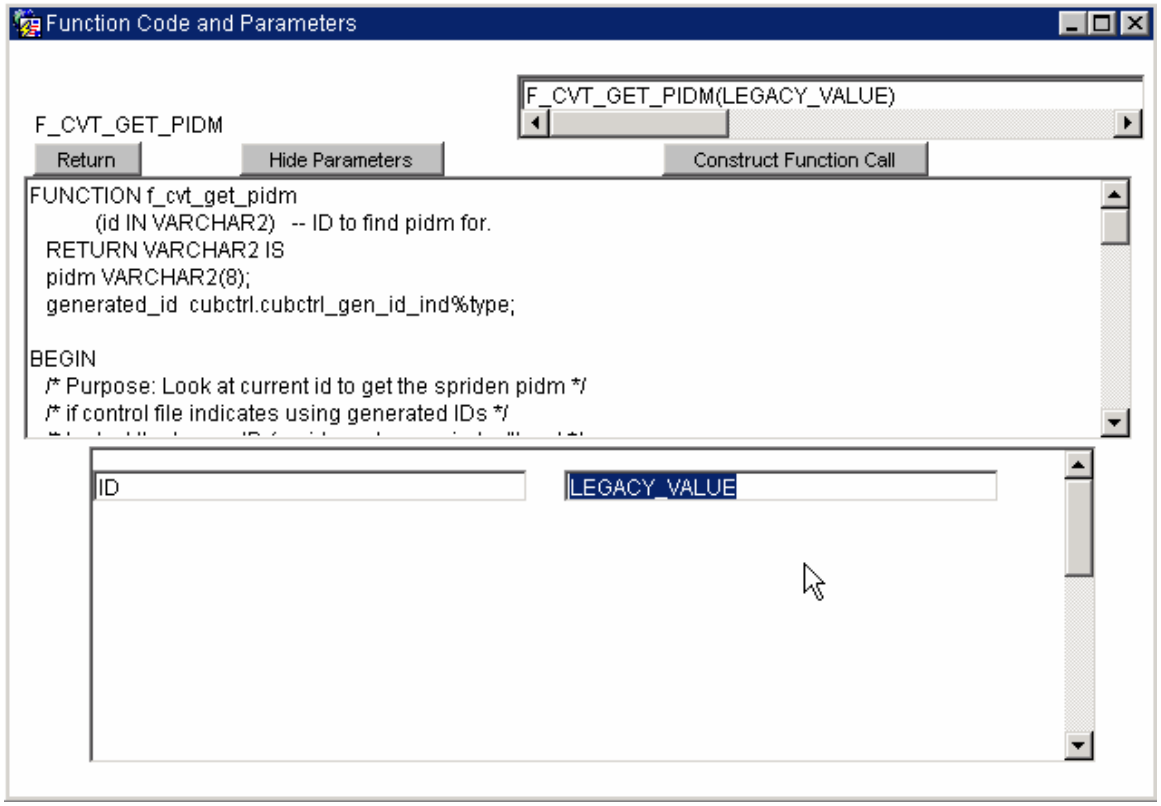
When the window is first opened, the text of the function will appear. The buttons across the top are RETURN, EDIT PARAMETERS, and CONSTRUCT FUNCTION CALL.

When you click the EDIT PARAMETERS button, you will see the dialog box shown in the image, and the EDIT PARAMETERS button changes to Hide PARAMETERS, as shown in the following image.

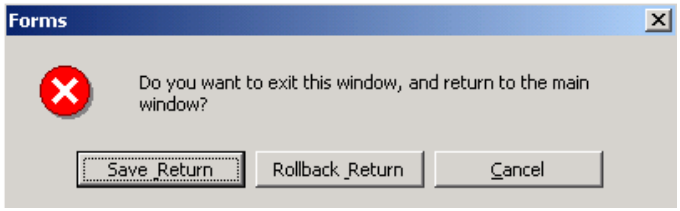


Parameter names default into the left column of the parameters dialog box. Enter the parameter value in the right column. In this function, F_CVT_GET_PIDM, the value that must be passed to the function is the legacy value in the *convert* column of the convert table. The correct variable, already defined in the convert script, is LEGACY_VALUE. For a more detailed discussion of the construction of the convert table and the contents of the columns in the convert table, please see the "Scripts Produced by the SCT Converter Tool" section of this document.

When you have entered the correct parameter, click the CONSTRUCT FUNCTION CALL button. The box on the top right corner of the form is populated with the function call, as in the image below.



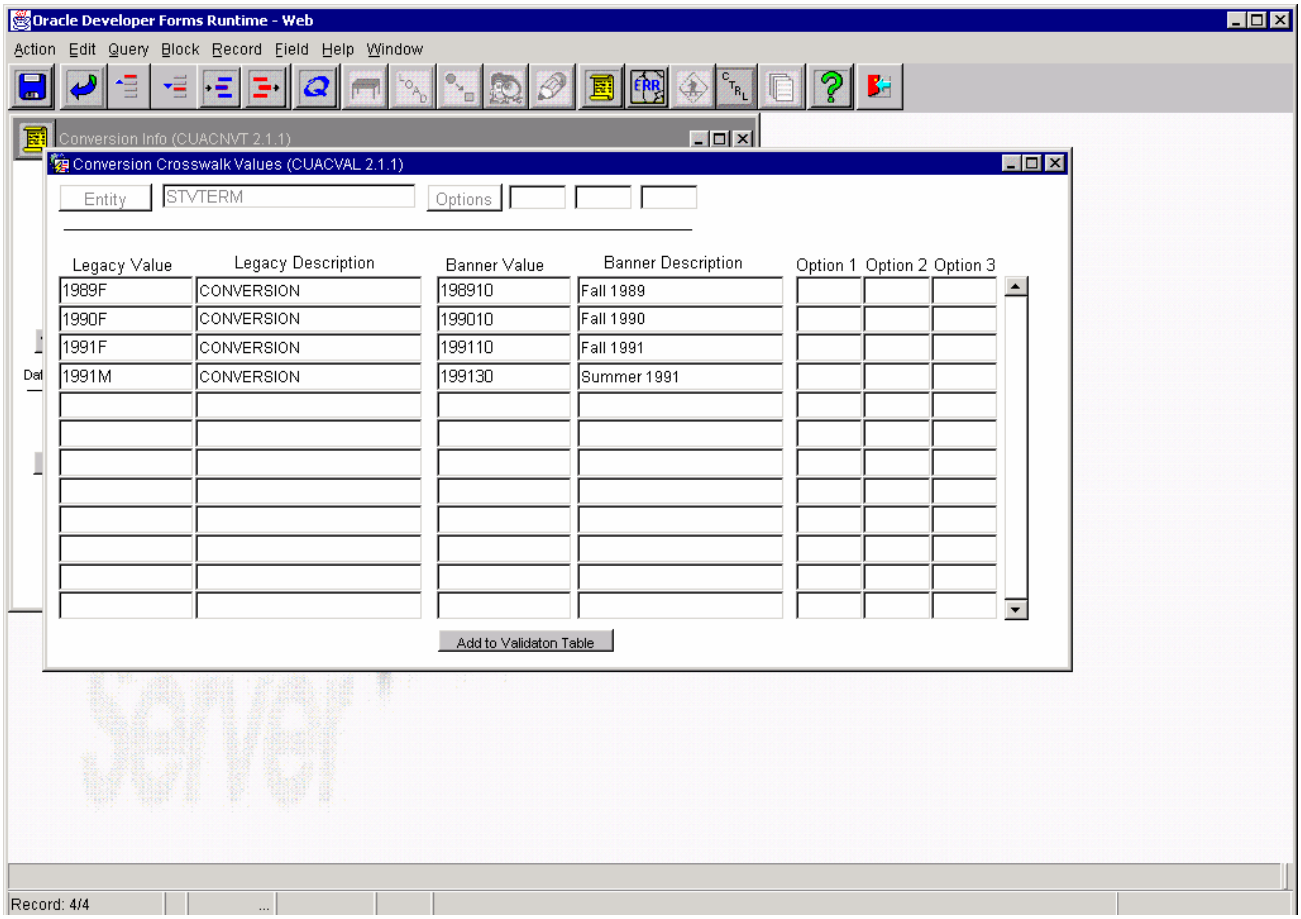
After constructing the function call, click the RETURN button. You will have three choices:



Choosing "Save & Return" will return your cursor to the Table Block of CUACNVT. You must perform a NEXT BLOCK function to Navigate to the column block and resume data entry. For more detailed information about the functions delivered with the Converter Tool, review the on-line documentation for each function or review the same function documentation in the cvt_genlib.sql file in the install directory of the Converter Tool.

Using the Crosswalk Feature of the Converter Tool: CUACVAL

Clicking the CROSSWALK button on the Tool Bar provides access to the "Conversion Crosswalk Values" window, which allows set up of an equivalency table between legacy values and SCT Banner values.



The "Entity" field allows you to give a name to the crosswalk grouping. This entity name is user-defined and is used as a parameter in the function call for the F_CVT_CURCVAL_RNULL function, which will use the crosswalk to insert the SCT Banner values into the appropriate <*TargetTable*> columns during the conversion process Naming convention for Crosswalk Entities: The crosswalk entity should be named after the target table for which the crosswalk is being created. For example, to crosswalk address types for the STVATYP validation table, the crosswalk should be named "STVATYP." To crosswalk detail codes for the TBBDETC table, the crosswalk entity should be named "TBBDETC."

If there is a one-to-one relationship between legacy values and SCT Banner values, you will simply enter the legacy values in the "Legacy Value" column and the equivalent SCT Banner values in the "SCT Banner Value" column. You would then SAVE , exit the crosswalk, return to CUACNVT and use the F_CVT_CURVAL_RNULL function on the appropriate field (See the following section "TO USE THE CROSSWALK VALUES IN THE CONVERSION SCRIPT").

The Legacy Description field can be loaded if conversion data from the legacy system is supplied. If the cvt_chkxwlk.sql utility script is used after data is loaded into a _CVT temporary table, any values for columns using a crosswalk function will be loaded into the CUACVAL form (CURCVAL table) with a Legacy description of CONVERSION and a SCT Banner description of CONVERSION. The SCT Banner description is used only to populate the description field for validation tables as described in the following paragraph.

When the Entity in the Key block is a SCT Banner validation table and there is a one-to-one relationship between the Legacy Value and SCT Banner Value columns, you can use the Add to Validation Table button to add any new values to the validation table that are listed in the crosswalk form but are not in the validation table.

Using Options on the Crosswalk Form

If it is necessary to use multiple values from your legacy system to translate to one SCT Banner value, you will need to use the option columns on the crosswalk. Your flat file will need to include all values necessary for the translation. If any of the values in the flat file do not correspond to a column in the SCT Banner table, you will need to add a column or columns to the <target_table> secifications on the CUACNVT form.

An example of a situation that might call for use of the crosswalk options columns is in the translation of legacy address codes into the SCT Banner SPRADDR address table.

An institution may have several campuses, each of which has had its own different address codes that indicate a student's permanent address, but for the new system, those address codes need to be standardized across campuses. Therefore, the campus code is needed in order for a correct translation of address code values to occur.

Step One -- Preparing the Flat File:

The flat file used as the data source file for the SPRADDR table must include the campus code. For this example, assume the campus code follows the address type in the legacy file.

Step Two -- Set up Specifications on CUACNVT:

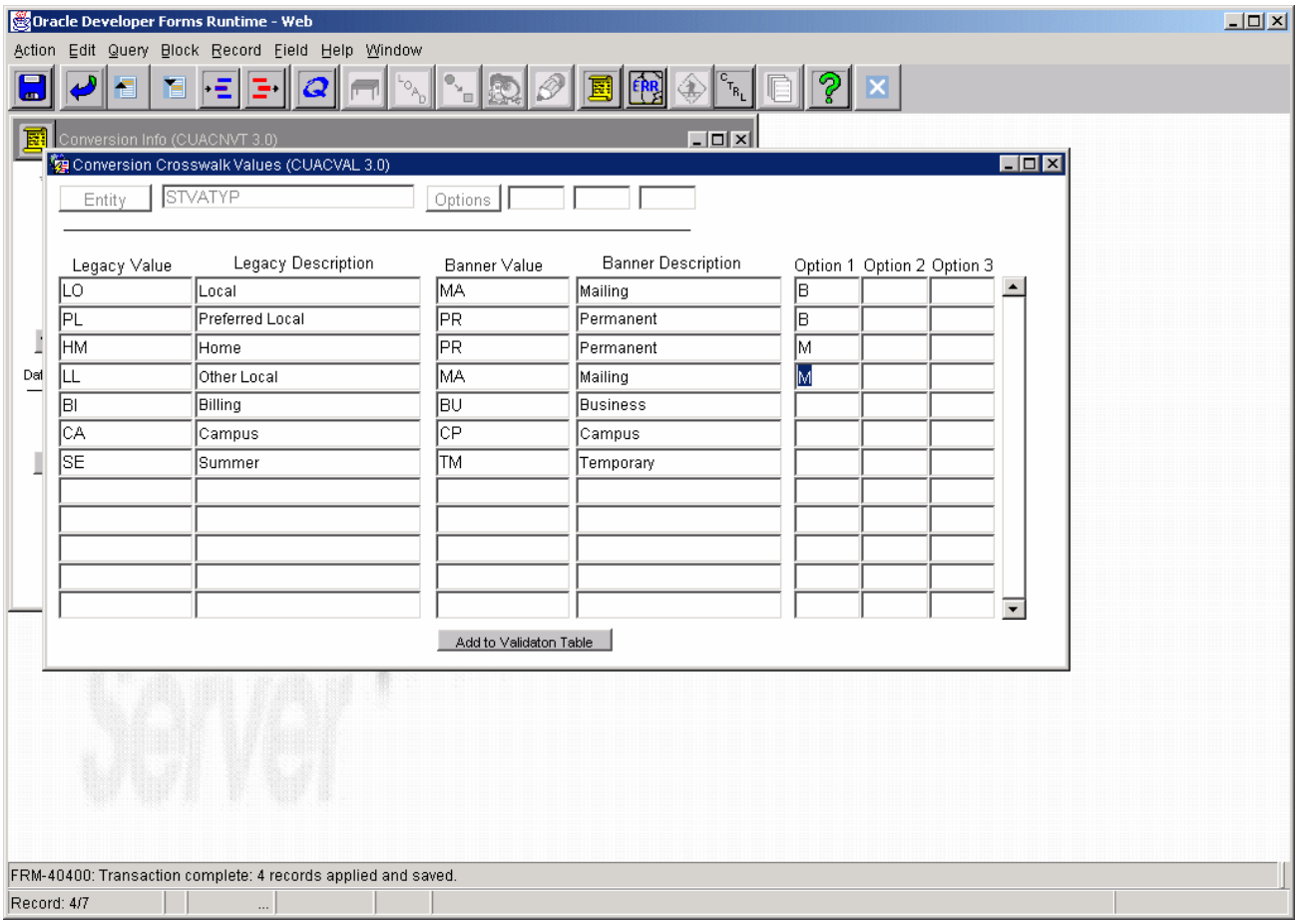
On the CUACNVT form for the SPRADDR table, you will insert a column after the *spraddr_atyp_code,* name it *spraddr_camp_code,* and enter the correct length, etc. Check the "load" box, but *do not* check the "insert" box. You will only be loading legacy data into the convert table, SPRADDR_CVT, so that these campus values can be passed as parameters to the F_CVT_CURVAL function and used in the translation of address types. You do not load this data into the target table.

Step Three -- Set up the Crosswalk Form (CUACVAL)

Name your entity STVATYP. Do not enter any data in the Options fields in the Key Block at this time. These fields are used as query fields after the crosswalk has been completed to allow you to see all specifications for a particular value in any option column.

NEXT BLOCK

Enter the legacy value that you wish to translate, the SCT Banner value, and in the Option 1 Column, enter the first campus value that is required for a correct translation of the address code. Add a row for each address code + campus code combination. For this example, the correct setup of the crosswalk table is displayed in the following image.



Notice the two sets of records which have options. In each set, the campus codes of “B” and “M” are used to set the extra condition needed for an accurate translation of values into SCT Banner.

Using Crosswalk Values in the Conversion Script

With your cursor in the "Convert Fnctn" field, press the CONVERT FNCTN button and Choose F_CVT_CURCVAL_RNULL.

Click the VIEW FUNCTION button on the Tool Bar. In the "Function Code and Parameters" window, press the EDIT PARAMETERS button.

Enter the following parameters, using the STVATYP entity from the image:

CONVERSION_ENTITY = "STVATYP" (or the name of your entity that you have created – *must be enclosed in single quotes*)

LEGACY_VALUE = LEGACY_VALUE

Note: Any parameter passed that is not a defined variable must be enclosed in single quotation marks. In this example, the name of our entity is enclosed in quotes, and the variable LEGACY_VALUE is not enclosed in quotes.

To include options as parameters, you must identify the option with the name that appears in the convert script. That naming convention is:

<*target_table*>_rec.convert_column_name [the name of the column that holds the value that is used as the "option".

In our example of the STVATYP entity, which is being used to translate values for the SPRADDR table, the parameter for Option1 would be:

spraddr_rec.convert_camp_code

Press the CONSTRUCT FUNCTION CALL button.

Press the RETURN button. Choose "Save & Return."

NEXT BLOCK to the Column Block of the form and continue to set up specifications.

Note: The naming convention for the option parameters is derived from the name assigned to the convert column in the convert script. It is identified by the cursor name and the column name. To view convert script code, refer to Appendix D: Sample Conversion Scripts.

Loading Data into the SPRIDEN Table

The SCT Banner SPRIDEN table is the first table to be loaded with person data. The SCT Banner PIDM is created during the loading of the SPRIDEN table. The SCT Converter Tool function, F_CVT_AUTO_PIDM, generates the PIDMs and dynamically updates the SOBSEQN sequence table after the assignment of each new pidm. This pidm-generating function assumes a specific data configuration for the legacy data in the spriden_cvt.dat file. The flat file for loading spriden must follow the designated configuration below in order for the F_CVT_AUTO_PIDM to correctly assign pidms to incoming legacy name/identification records.

Required file layout for SPRIDEN_CVT.DAT:

CONVERT_PIDM – Null if record is the current record. If not current record because of name or ID change, this field should contain the current legacy ID, and the convert_change_ind column should contain the appropriate indicator: I if the record represents an ID change, and N if the record indicates a name change

CONVERT_ID – Current ID if record is current. If not current ID, this field should contain the previous ID and the spriden_change_ind field should contain "I".

CONVERT_LAST_NAME – Current last name if last name is current. If not current last name, this field should contain previous last name, and the spriden_change_ind field should contain "N".

CONVERT_FIRST_NAME– Current first name if first name is current. If not current first name, this field should contain previous first name, and the spriden_change_ind field should contain "N".

CONVERT_MI– Current middle name if middle name is current. If not current middle name, this field should contain previous middle name, and the spriden_change_ind field should contain "N".

CONVERT_CHANGE_IND – NULL if record is current. If not current record, this field should contain "I" if the ID is not current, and "N" if the name is not current.

CONVERT_ENTITY_IND – P if record is a person record; C if record is a non-person record.

CONVERT_ACTIVITY_DATE – optional for data file. Can be legacy maintenance date or can be inserted as sysdate or default date during conversion.

CONVERT_USER – Should be some identifiable conversion user, such as "CONVERT." Typically populated with default value using Converter Tool default value field.

CONVERT_ORIGIN – optional. Typically populated with default value of "CONVERSION" using Converter Tool default value field.

Sample spriden_cvt.dat file layout:

```
DATA:.....
,111111111,SMITH,AMELIA,KAY,,P
111111111,101101101,SMITH,AMELIA,KAY,I,P
          ,222222222,SIMONEAUX,MARYANNE,FRANCIS,,P
222222222,222222222,CAMPESI,MARYANNE,S,N,P
          ,333333333,WOODRUFF,LUCILLE,MARIE,,P
          ,444444444,CAMPBELL,JOSEPH,ALEXANDER,,P
          ,555555555,PITTMANN,MILDRED,YATES,,P
          ,666666666,BOYD,WILLIAM,GUERRY,,P
          ,777777777,GLOVER,SAVION,TAP,,P
          ,888888888,WINGFIELD,WILLIAM,DOUGLASS,,P
          ,999999999,LENNON,JOHN,O,,P
          ,000000000,ELDRIDGE,GROVER,BLUE,,P
```

Generating SCT Banner PIDM's

To generate SCT Banner pidms for new records, users must use the F_CVT_AUTO_PIDM function. This function generates the SCT Banner PIDMs and dynamically updates the SOBSEQN sequence table after the assignment of each new pidm. In addition to using the required spriden_cvt.dat file layout, users must edit the spriden_convert.sql script to re-order the data in the cursor. This is the only occasion in which manual editing of any of the converter tool scripts is required.

If a NULL convert_pidm is passed to the function, then a pidm will be generated from SOBSEQN since this will signify the most current record. If convert_pidm has a value (this means either a NAME change or an ID change is in the extract), then the pidm should not be generated from SOBSEQN. Rather, the pidm will be fetched from the current convert table, spriden_cvt, where the convert_pidm is equal to the convert_id and the convert_change_ind is null.

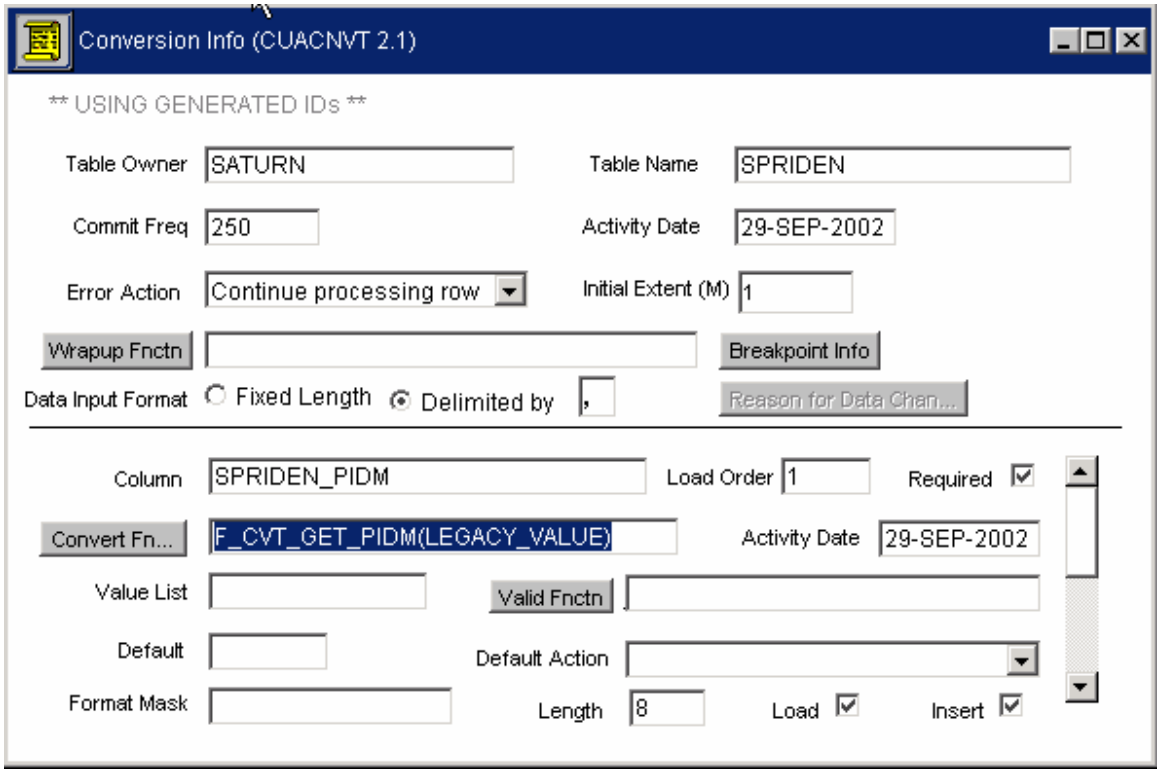
In order for this method to work, the current record must be processed first by the conversion program spriden_convert.sql (meaning: the cursor must have an 'ORDER BY' clause so that records with null change indicators are processed first. ex: ...order by convert_change_ind desc; . . .).

Steps for loading the SPRIDEN table using generated PIDMs and generated IDs are outlined in the next section of this document. Further, steps for loading the SPRIDEN table are included in the Sample Conversions section of this document.

Using the Generated ID Feature of the Converter Tool

The Control Form provides the ability to check whether or not you are using Generated ID's.

If the "Use Generated IDs" check box is marked and saved, upon returning to the main converter tool form, CUACNVT, a message is displayed at the top of the form to indicate the use of Generated ID's.



The Generated ID Feature includes three parts:

- Converter Tool Control Form
- Convert Script for SPRIDEN
- F_CVT_GET_PIDM function

Use the following steps:

1. Mark the "Use Generated IDs" indicator on the Conversion Tool Control Form (CUACTRL) and save your work.
2. Set up SPRIDEN conversion and load data into the SPRIDEN_CVT table.
3. Create the SPRIDEN_CONVERT script (should contain the necessary code to generate all expected ID's).

4. Change the "order by" clause for the spriden_cursor FROM "order by spriden_cvt_record_id" TO "order by convert_change_ind desc Edit the spriden_convert script as follows:

```
:
CURSOR spriden_cursor IS
    SELECT * FROM spriden_CVT
     WHERE spriden_cvt_status = records_in
~~order by spriden_cvt_record_id;~~
order by convert_change_ind desc;
```

5. Ensure the "LGCY" validation value exists in the GTVNTYP table. If it does not exist, add it as new validation value where the code should be set to "LGCY" and the description can be set to "Legacy ID" (or any other desired description).
6. Execute the SPRIDEN_CONVERT.SQL script to convert all Name and ID information in SPRIDEN_CVT.
7. Execute the SPRIDEN_CONVERT.SQL script to take rows from SPRIDEN_CVT and insert them into SPRIDEN.
8. Convert any remaining data.

Operational Concepts

SPRIDEN_NTYP_CODE – Gives information about the SPRIDEN record. When the SPRIDEN_NTYP_CODE is "LGCY", that indicates the SPRIDEN record is the legacy ID.

SPRIDEN_CHANGE_IND – Used to indicate how the ID record was changed. If the value in this column is NULL, the record is the current Name and ID. The other standard SCT Banner values are "N" for a name change and "I" for an ID changes. For purposes of conversion, we are using this column with a value of "I" in conjunction with the SPRIDEN_NTYP_CODE having a value of "LGCY" to indicate the legacy ID. Because the SPRIDEN_CHANGE_IND column will not be null, the legacy ID will not be the current ID, the generated ID will be. This is used in the F_CVT_GET_PIDM function (explained later).

SOBSEQN table – Contains sequence numbers that are used for various things in SCT Banner. ID and PIDM are two numbers stored in this table. When ID's are generated, you get the next available ID number in SOBSEQN The F_CVT_GET_PIDM function dynamically updates the SOBSEQN_MAXSEQNO where the SOBSEQN_FUNCTION = "PIDM".

SOBSEQN_SEQNO_PREFIX – The leading character that is used for generated ID's. The initial value is "@". Change this value if you want generated ID's to have a different leading character.

Convert Script for SPRIDEN

If the "Using Generated IDs" check box is marked on the Conversion Tool Control Form, the first step is to convert all name and ID information into SPRIDEN. The conversion process is the same whether or not you are using Generated ID's.

1. Set up your SPRIDEN conversion in the converter tool.
2. Create your SPRIDEN_CVT table.
3. Load data into the SPRIDEN_CVT table.
4. Create your SPRIDEN_CONVERT script (should contain the necessary code to generate your ID's).

5. Modify the spriden_convert.sql script to change the "order by" clause in the cursor statement:

```
CURSOR spriden_cursor IS
   SELECT * FROM spriden_CVT
    WHERE spriden_cvt_status = records_in
    order by spriden_cvt_record_id;
   order by convert_change_ind desc;
```

Note: There is one restriction on converting SPRIDEN records with Generated ID's. You must run the conversion script generated by the converter tool in two separate steps. The conversion script allows you to convert records in SPRIDEN_CVT and insert them into SPRIDEN in one step (disposition of "B"). When using the Generated ID feature, you must run the script with the Convert disposition and then run the script with the Insert disposition. In other words, you cannot use the action that allows you to both convert and insert in the same step (disposition of "B").

The reason has to do with how the Generated ID's are created. The process to create SPRIDEN records (with Generated ID's) works after all the records have been converted and only creates a Generated ID record for those SPRIDEN_CVT records with a status of "C" that don't have a Generated ID. This information can only be established after the records are converted and before they are inserted into SPRIDEN.

How It Works

After the SPRIDEN_CONVERT.SQL script converts all data in the SPRIDEN_CVT table, all records with a status of "C" (converted, as opposed to inserted (I) or errored (E)) are taken, where the change indicator (SPRIDEN_CVT.SPRIDEN_CHANGE_IND) is null. This is the current ID from the legacy system. The script sets the SPRIDEN_CVT.SPRIDEN_CHANGE_IND (change indicator) to "I" and the SPRIDEN_NTYP_CODE to "LGCY" to indicate this is the most current legacy ID. The script then gets the next available Generated ID number from the SOBSEQN table and creates a new SPRIDEN_CVT record using the Generated ID and copies all information from the converted legacy ID record. When the SPRIDEN_CONVERT.SQL script is completed, you are left with the legacy ID in the SPRIDEN_CVT table. The change indicator is set to "I". SPRIDEN_NTYP_CODE of "LGCY" (to be used by the F_CVT_GET_PIDM function) and a new Generated ID record in the SPRIDEN_CVT table is also produced with the same pidm, but the change indicator is NULL.

A Sample Run

The listing below shows a sample run of the SPRIDEN_CONVERT script using the Generated ID feature.

```
Please enter the record type which should be processed.
(N)ew records
(C)onverted records
(E)rrored records
Include:N
```

```
Please enter the disposition to which the records should
be processed.
(C)onvert records only
(I)nsert records only (records were previously converted successfully).
(B)oth convert and records into spriden
Disposition: C

================================================
= will be creating generated id records       =
= in SPRIDEN_CVT for any converted records    =
=  (i.e. records with spriden_cvt_status = 'C' =
================================================

Beginning Conversion of Table spriden
Job Number: 66
Number of ids generated = 264
Number of Rows Converted in SPRIDEN_CVT: 264
Number of Rows Inserted into spriden: 0
Number of Rows That Errored: 0

Completed Processing of spriden

PL/SQL procedure successfully completed.

Commit complete.
```

When the SPRIDEN_CONVERT.SQL script is executed a second time (to insert the converted records into the actual SCT Banner SPRIDEN table), two ID records are created for each person:

- The legacy ID that has the change indicator of "I" and the SPRIDEN_NTYP_CODE of "LGCY"
- The Generated ID with the change indicator of NULL

In the following example (that continues from the previous example), twice as many SPRIDEN records are loaded (528) than were originally loaded into SPRIDEN_CVT (264). This is because a Generated ID record for every current legacy ID record was created.

```
Please enter the record type which should be processed.
(N)ew records
(C)onverted records
(E)rrored records
Include:C

Please enter the disposition to which the records should
be processed.
(C)onvert records only
(I)nsert records only (records were previously converted successfully).
(B)oth convert and records into spriden
Disposition: I
```

```
================================================
= will be creating generated id records       =
= in SPRIDEN_CVT for any converted records    =
=  (i.e. records with spriden_cvt_status = 'C' =
================================================

Beginning Conversion of Table spriden
Job Number: 67
Number of ids generated = 528
Number of Rows Converted in SPRIDEN_CVT: 0
Number of Rows Inserted into spriden: 528
Number of Rows That Errored: 0

Completed Processing of spriden

PL/SQL procedure successfully completed.

Commit complete.
```

Note: If non-current ID's are being loaded for historical information in addition to the most current legacy ID, the number of records produced will not be exactly twice the number of records loaded into SPRIDEN_CVT.

Multiple Iterations of Converting SPRIDEN

Converting name and ID information into SPRIDEN from one system is relatively straightforward.

1. Load SPRIDEN_CVT and run the SPRIDEN_CONVERT.SQL script.
2. Fix any errors along the way and everything is in SPRIDEN.

But how is the SCT Converter Tool used to convert name and ID information from multiple systems, especially when the same people can be found in multiple systems. For example, some students are included in the Human Resource system as employees but will also be found in the Student system. If you need to convert your Human Resource system first, how do you only create SPRIDEN records for those students who are not in the Human Resource system (i.e. have not been put into SPRIDEN yet).

There are two things to change in the SCT Converter Tool on the SPRIDEN table rules. First, change the Convert Function for SPRIDEN_PIDM. Change this function to reflect:

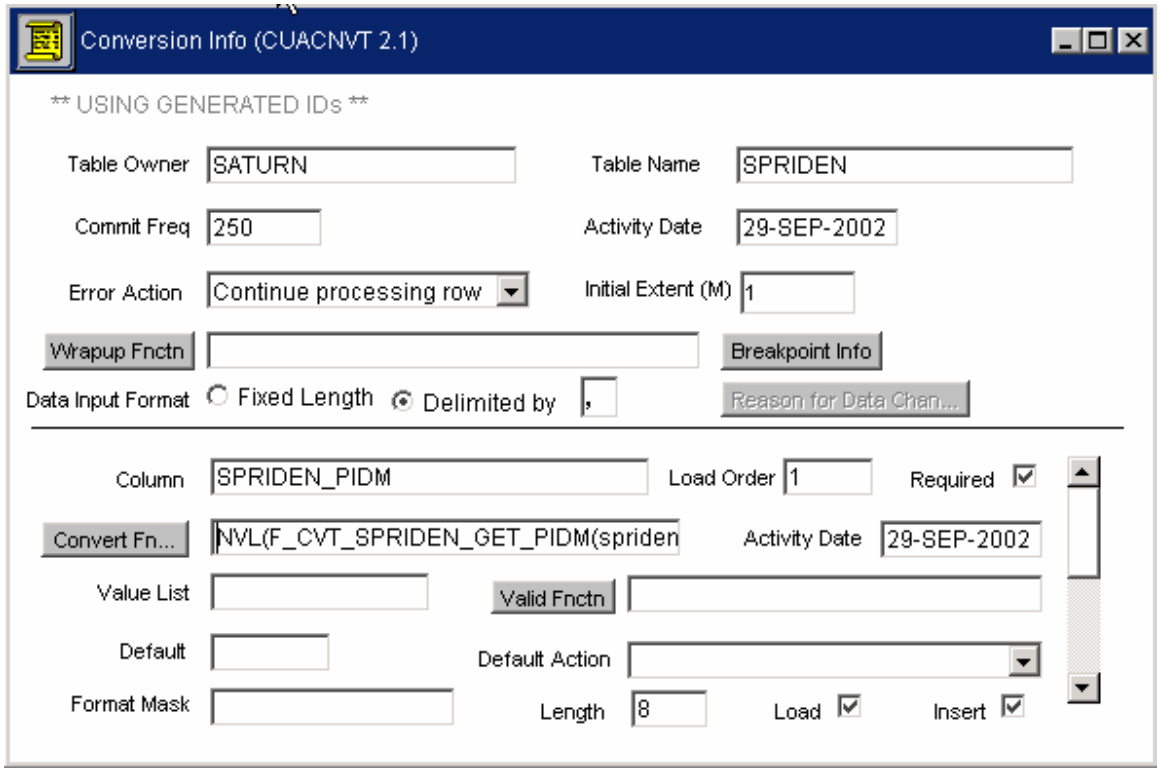
```
NVL(F_CVT_SPRIDEN_GET_PIDM(spriden_rec.convert_id),F_CVT_AUTO_PIDM(LE
GACY_VALUE));
```

Nesting these functions within the Oracle NVL function says, if:

F_CVT_SPRIDEN_GET_PIDM returns a NULL value (i.e. no matching ID found in the SCT Banner SPRIDEN table), then use F_CVT_AUTO_PIDM to create a new pidm. So after the SPRIDEN_CONVERT.SQL function is run, the SPRIDEN_PIDM column in the SPRIDEN_CVT table will have a pidm that is either an existing pidm from the SCT Banner SPRIDEN table (F_CVT_SPRIDEN_GET_PIDM) or a new pidm (F_CVT_AUTO_PIDM).

F_CVT_SPRIDEN_GET_PIDM is a new function with the 2.0 version of the tool. It is different from the original F_CVT_GET_PIDM because it returns a NULL for the pidm instead of an error when looking for a pidm in the SCT Banner SPRIDEN table. All other tables should use F_CVT_GET_PIDM to retrieve the PIDM from SPRIDEN, because in those cases, you do want an error if the ID has not been converted.

The other step in converting name and ID records into SPRIDEN when the name and ID already exist is to use a new wrap up function F_WRAP_SPRIDEN_EXISTS. This function is put in the Wrap Up Function field in the top half of the CUACNVT form. This function executes after all records have been converted and the status in the SPRIDEN_CVT record is “”C”. The function looks at the converted records (SPRIDEN_CVT_STATUS = “C”) and if the pidm in the convert table (SPRIDEN_CVT) exists in the SCT Banner Spriden table, it sets the status to “X” so that the SPRIDEN record is not inserted into SCT Banner.



Getting the PIDM for Other Tables using F_CVT_GET_PIDM

With new ID’s created for all persons from the legacy system into SCT Banner, the question is “How do I relate the Generated ID record back to the legacy ID for the rest of the data when I need a pidm?”. The answer is the F_CVT_GET_PIDM function.

The F_CVT_GET_PIDM function is used regardless of Generated ID’s. This function behaves differently depending on how the “Use Generated IDs” parameter is configured.

“Use Generated IDs” not marked

The F_CVT_GET_PIDM function takes the ID passed to it and looks for:

- A matching SCT Banner SPRIDEN ID record

  and

- A NULL change indicator (i.e. the most current record).

"Use Generated ID's" marked

The F_CVT_GET_PIDM function takes the ID passed to it and looks for:

- A matching SCT Banner SPRIDEN ID record

  and

- A change indicator set to "I"

  and

- The SPRIDEN_NTYP_CODE set to "LGCY"

This is why the SPRIDEN_CONVERT.SQL script takes the legacy ID record in SPRIDEN_CVT and sets the change indicator to "I" and the SPRIDEN_NTYP_CODE to "LGCY". A change indicator of "I", and the SPRIDEN_NTYP_CODE of "LGCY", provides a way to easily find the PIDM that is associated with the legacy ID (even though a Generated ID was created for the legacy ID).

Chapter 3 - A Conversion Example Using the SCT Converter Tool

Establishing Specifications for Conversion of SPRIDEN Table 3-5
Using CUACNVT to produce the <TargetTable>_cvt_create.sql script 3-11
Using CUACNVT to produce the <TargetTable>cvt.sql script..... 3-12
Using CUACNVT to produce the <TargetTable>convert.sql script..... 3-13
Establishing Specifications for Conversion of SPRADDR Table 3-14
Establishing Specifications for Conversion of SPBPERS Table 3-20

A Conversion Example Using the SCT Converter Tool

This section examines the steps of a sample conversion, using CUACNVT to create the necessary scripts to accomplish the process.

There are examples for loading data into three SCT Banner tables where shared data resides for all products:

SPRIDEN (Identification Table)

SPBPERS (Person Biographic/Demographic Information Table)

SPRADDR (Address Table)

The flat files that contain the legacy data (spriden_cvt.dat, spbpers_cvt.dat, and spraddr_cvt.dat) are included in Appendix A, Sample Data Files Used in Examples.

Assumptions:

Users will be generating new SCT Banner ID's.

Spriden_cvt.dat data file will be configured according to guidelines outlined in Section 2-18 of this user manual.

Establishing Specifications for Conversion of SPRIDEN Table

The following steps will show how to use the SCT Converter Tool to set up specifications for converting data into the SCT Banner SPRIDEN table:

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Open CUACNVT and log in. Login: your_convert_username Password: your_convert_user password Database: your_database	CUACNVT	
Perform an INSERT RECORD function.	CUACNVT Table Block	
Enter the name of the table owner, SATURN.		If you are practicing, or if you are in training, the owner of your table may be your user name, your login, or some other owner. If you are setting up specifications for your conversion, use the SATURN owner.
Navigate to the Next Field.		
Enter the name of the table, SPRIDEN.		
Navigate to the "Error Action" field.		Accept the default Error Action, "Continue Processing Row."
Navigate to the "Delimited By" field and insert a comma.		For this exercise, our data is comma-delimited. For your conversion, you have the option of using comma-delimited data files or fixed length data files. This field should be populated to reflect the file layout of your data.
SAVE your changes		As you save changes and Navigate to the next block, the column block is populated with columns from the SPRIDEN table
NEXT BLOCK	CUACNVT Column Block	To see all columns in the table, after moving to the Column Block, with your cursor in the "Column" field, move through the records with the NEXT RECORD button or by using the down arrow.
Navigate to the SPRIDEN_PIDM column.		
Navigate to the "Convert Fnctn" field and click the CNVERT FNCTN button.		
SELECT "F_CVT_AUTO_PIDM" from the list.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
With your cursor in the "Convert Fnctn" field, press the VIEW FUNCTION button on the Tool Bar.	Function Code and Parameters Window	The View Function icon is the icon to the left of the Pencil. Please be patient. It takes a few seconds for this window to appear. You will see the "Function Code and Parameters" window. It shows the text of the function create script, and has three buttons across the top of the window above the function text. The F_CVT_AUTO_PIDM function will generate a pidm if a pidm is not passed to the function. Note that later in this process you must manually edit the spriden_convert.sql script to change the "order by" clause in the spriden_cursor statement.
Press the EDIT PARAMETERS button.	Edit Parameters Dialog Box	Notice in the text of the function that there is only one parameter: PIDM. The left side of the parameters dialog box has the word "PIDM" defaulted in.
Enter LEGACY_VALUE in the right column of the parameters dialog box.		LEGACY_VALUE is a variable that has been defined in the convert script generated by the converter tool. It refers to the legacy data value in the convert column in the convert table. Any time it is used as a parameter in a function, the actual value that is passed is the legacy data value that resides in the convert column that corresponds to the target table column to which the function is being applied and to which to data will be moved in the conversion process. Not case sensitive.
Press the CONSTRUCT FUNCTION CALL button.		When you press the CONSTRUCT FUNCTION CALL button, the blank box at the top right corner of the form is populated with the actual function call that will appear in the convert script: F_CVT_AUTO_PIDM(LEGACY_VALUE)
Press the RETURN button.		
Choose "Save & Return."		Doing this will return your cursor to the Table Block of CUACNVT. To resume work, you must perform a NEXT BLOCK function.
NEXT BLOCK	CUACNVT	
Navigate to the Length field. Change the value from "8" to "9."		This action will increase the size of the CONVERT_PIDM column in the convert table from 8 characters to 9 characters. This is necessary to accommodate the 9 character legacy ID values that reside in the CONVERT_PIDM positions in the date file.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the "Load" field in the Column Block and verify that it is *checked*.		Selecting the LOAD button creates a line in the SQL*Loader script to indicate that there is legacy data in the flat file for this column. Please review the "Generating SCT Banner PIDMs" section of this document for a detailed explanation of the required file layout for the spriden_cvt.dat file. In the sample data file, there are records with values in the convert_pidm column.
Navigate to the "Insert" indicator in the Column Block and check the "Insert" indicator.		Checking the INSERT button indicates that you want this column to appear in the conversion script to be converted to the target table columns in the convert table and subsequently inserted into the actual target table. If you have edited out the unnecessary columns in the beginning of this process, you will check "Insert" for all remaining columns. If you choose not to edit out unnecessary columns in the beginning of this process, you may indicate with the "Insert" indicator whether or not they are to be converted/inserted.
Navigate to the Column field and perform a NEXT RECORD function to display the SPRIDEN_ID column.	CUACNVT Column Block	
Navigate to the Load field and check the box.		Checking the LOAD box indicates that you want this column to appear in the SQL*Loader script because you have legacy data in the flat file for this column to be loaded into the convert table.
Navigate to the Insert field check the box.		Checking the INSERT box indicates that you want this column to appear in the conversion script to be converted to the target table columns in the convert table and subsequently inserted into the actual target table.
Navigate to the next record, SPRIDEN_LAST_NAME		
Navigate to the "Convert Fnctn" field.		All names in the legacy data file are in ALL CAPS. It is appropriate here to use Oracle's INITCAP function to change the three SPRIDEN name fields to mixed case.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Enter the following: INITCAP(LEGACY_VALUE)		This will cause the legacy data to change from ALL CAPS to Mixed Case during the conversion process. By using the LEGACY_VALUE variable here, the INITCAP function will be applied to all legacy data values in the CONVERT_LAST_NAME field as they are moved to the SPRIDEN_LAST_NAME field during the conversion process. For a look at the details of the spriden_convert.sql script, refer to Appendix D: Sample Conversion Scripts.
Navigate to the Load and Insert Options and check both.		
Navigate to the Next Record, SPRIDEN_FIRST_NAME		
Navigate to the “Convert Fnctn” field.		
Enter the following: INITCAP(LEGACY_VALUE).		
Navigate to the “Convert Fnctn” field.		
Enter the following: INITCAP(LEGACY_VALUE).		
Navigate to the Load and Insert Options and check both.		
Navigate to the Next Record, SPRIDEN_MI		
Navigate to the “Convert Fnctn” field.		
Enter the following: INITCAP(LEGACY_VALUE)		
Navigate to the Load and Insert Options and check both.		
Navigate to the Next Record, SPRIDEN_CHANGE_IND		
Navigate to the Load and Insert options and check both.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the Next Record, SPRIDEN_ENTITY_IND		This is one of the fields for which you can enter a default value of "P". Our sample data contains the "P" data value for all records. In your conversion, if you are certain that all records you are loading are person records, you may default the "P" entity indicator here, or you may include it in the data file. If it is defaulted in the Converter Tool, the LOAD box will be unchecked and the INSERT box will be checked.
Navigate to the Load and Insert options and check both.		Note: If in your conversion, you default a value for spriden_entity_ind, you will uncheck LOAD and check INSERT.
Navigate to the Next Record, SPRIDEN_ACTIVITY_DATE.		
Navigate to the Convert Function field and enter SYSDATE.		This will populate the spriden_activity_date field with the current system date. If during your conversion you wish to load the legacy system maintenance date into this field, you would leave the Convert Function field null and insert the date format mask into the Format Mask field. For example, if your legacy date is YYYYMMDD, you would insert this format mask into the Format Mask field.
Navigate to the Load and Insert fields. Uncheck LOAD, Check INSERT.		No legacy date exists for this column, but you do want the presecribed value, SYSDATE, inserted during the conversion.
Navigate to the next record, SPRIDEN_USER.		
Navigate to the "Default" field		
Enter SCTCVT		No quotation marks.
Navigate to the "Default Action" column.		
Choose "(D)efault if no legacy passed."		You may also choose "(O)verride any legacy passed."
Navigate to the Load option and verify that it is unchecked.		No legacy data exists for this column.
Navigate to the Insert option and check it.		
Navigate to the Next Record, SPRIDEN_ORIGIN		
Navigate to the "Default" field		
Enter CONVERSION		No quotation marks.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the "Default Action" column.		
Choose "(D)efault if no legacy passed."		You may also choose "(O)verride any legacy passed."
Navigate to the Load option and verify that it is unchecked.		No legacy data exists for this column.
Navigate to the Insert option and check it.		
Navigate through the remainder of the SPRIDEN columns and Uncheck both LOAD and INSERT for all.		SPRIDEN_SEARCH and SPRIDEN_SOUNDEX fields are populated via a database trigger on the SPRIDEN table and do not need to be included in the LOAD or INSERT options for this table.
SAVE your changes.		

You have now entered all the necessary specifications for the SPRIDEN table conversion. The next step is to use the features of the converter tool to produce the following 3 scripts:

- spriden_cvt_create.sql – creates the convert table (SPRIDEN_CVT)
- spriden_cvt.ctl – used by SQL*Loader to load the data from the client's flat file (spriden_cvt.dat) to the convert table (SPRIDEN_CVT)
- spriden_convert.sql – moves the data from the convert columns to the target table columns within the convert table, then moves the data from the target table columns in the convert table to the corresponding columns in the actual target table in the database.

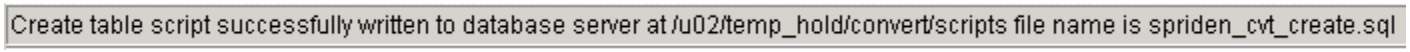
Steps in creating the scripts appear on the following pages.

Using CUACNVT to produce the *<TargetTable>_cvt_create.sql* script

With the cursor in the Table Block for the SPRIDEN table:

Click the CONVERT TABLE SCRIPT button: 

The following message appears on the message line (your database server name will differ from that in the example):

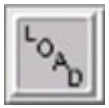


To view the text of the script, refer to Appendix B: Sample Table Creation Scripts.

Run this script in SQL on the server to create the convert table in the host database.

Using CUACNVT to produce the *<TargetTable>_cvt.ctl* script

With the cursor in the Table Block for the SPRIDEN table:

Press the SQL LOADER SCRIPT button: 

The following message appears on the message line (your database server name will differ from that in the example):

SQL*Loader control file script successfully written to database server at /u02/temp_hold/convert/scripts file name is spriden_cvt.ctl

To view the text of the script, refer to Appendix C: Sample SQL*Loader Scripts.

Be sure that your spriden_cvt.dat file is in the same directory on the server and run SQL*Loader to load the data into the convert table. Consult with your DBA for the correct command for executing SQL*Loader. A sample command:

sqlldr userid=your_convert_username/your_convert_user_password control=spriden_cvt.ctl

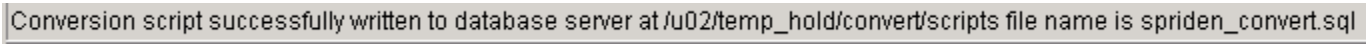
Using CUACNVT to produce the *<TargetTable>_convert.sql* script

With the cursor in the Table Block for the SPRIDEN table:

Press the CONVERSION SCRIPT button: 

Note: If you are planning to use the SCT Conversion Tool's Generated ID Feature, be sure to mark the "Use Generated IDs" checkbox on the Conversion Tool Control Form **before** creating SPRIDEN_CONVERT.SQL.

The following message appears on the message line (your database server name will differ from that in the example):



To view the text of the script, refer to Appendix D: Sample Conversion Scripts.

For the SPRIDEN table load (and only for the SPRIDEN table load) you will need to edit the spriden_convert.sql script to change the "order by" clause in the cursor statement. Please refer to the documentation for further instructions.

Once you have evaluated the data in the convert table *convert* columns, run this script in SQL on the server for the (N)ew records in (C)onvert mode. If there are errors, correct them, and run this script again, this time choosing (E)rrored records for the first parameter of the script and (C)onvert for the disposition. Continue this process until all errors are corrected. Once you are satisfied that all errors are correct and that your data is "clean," run this script in SQL on the server for the (C)onverted records in (I)nsert mode to insert the data into the SPRIDEN table.

Establishing Specifications for Conversion of SPRADDR Table

The following steps will show how to use the SCT Converter Tool to set up specifications for converting data into the SCT Banner SPRADDR table. Sample flat file appears in Appendix B: Sample Table Creation Scripts.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Open CUACNVT and log in. Login: your_convert_username Password: your_convert_user password Database: your_database	CUACNVT	
Perform an INSERT RECORD function.	CUACNVT Table Block	
Enter the name of the table owner, SATURN.		If you are practicing, or if you are in training, the owner of your table may be your user name, your login, or some other owner. If you are setting up specifications for your conversion, use the SATURN owner.
Navigate to the Next Field.		
Enter the name of the table, SPRADDR.		
Navigate to the "Error Action" field.		Accept the default Error Action, "Continue Processing Row."
Navigate to the "Delimited By" field and insert a comma.		For this exercise, our data is comma-delimited. For your conversion, you have the option of using comma-delimited data files or fixed length data files. This field should be populated to reflect the file layout of your data.
SAVE your changes		As you save changes and Navigate to the next block, the column block is populated with columns from the SPRADDR table
NEXT BLOCK	CUACNVT Column Block	To see all columns in the table, after moving to the Column Block, with your cursor in the "Column" field, move through the records with the NEXT RECORD button or by using the down arrow.
Navigate to the SPRADDR_PIDM column.		
Navigate to the "Convert Fnctn" field.		
Press the CONVERT FNCTN button.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Choose "F_CVT_GET_PIDM" from the list.		The assumption is that the legacy data has already been loaded into the SPRIDEN table. This function uses the legacy ID to match with SPRIDEN_ID. If a match is found, the existing SPRIDEN_PIDM is returned to populate the SPRADDR_PIDM field. If no match is found, the transaction will produce an error.
With your cursor in the "Convert Fnctn" field, press the VIEW FUNCTION button on the Tool Bar.	Function Code and Parameters Window	
Press the EDIT PARAMETERS button.	Edit Parameters Dialog Box	
Enter LEGACY_VALUE in the right column of the parameters dialog box.		In the *spraddr_cvt* table, the legacy value that will be loaded into the *convert_pidm* column is the legacy ID. That is the value that must be passed to the function. The name for that value is LEGACY_VALUE.
Press the CONSTRUCT FUNCTION CALL button.		When you press the CONSTRUCT FUNCTION CALL button, the blank box at the top right corner of the form is populated with the actual function call that will appear in the convert script: F_CVT_GET_PIDM(LEGACY_VALUE)
Press the RETURN button.		
Choose "Save & Return."		Doing this will return your cursor to the Table Block of CUACNVT. To resume work, you must perform a NEXT BLOCK function.
NEXT BLOCK	CUACNVT	
With your cursor in the SPRADDR_PIDM column field, Navigate to the "Length" field.		
Change the 8 to 9		You must change the length of the CONVERT_PIDM column in the table to accommodate the legacy ID, which is 9 characters long. The F_CVT_GET_PIDM function will use this ID to match against the existing SPRIDEN_ID, inserted previously, to get the pidm and insert it into the SPRADDR_PIDM field in the convert table and subsequently into the SPRADDR_PIDM field in the SPRADDR table.
Navigate to the "Load" field in the Column Block and check the "Load" indicator.		Checking the LOAD button indicates that you want this column to appear in the SQL*Loader script because you have legacy data in the flat file for this column to be loaded into the convert table.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the "Insert" indicator in the Column Block and check the "Insert" indicator.		Checking the INSERT button indicates that you want this column to appear in the conversion script to be converted to the target table columns in the convert table and subsequently inserted into the actual target table.
Navigate to the next record, the SPRADDR_ATYP_CODE column.	CUACNVT Column Block	
Click the Crosswalk icon to display the Crosswalk form (CUACVAL)		
Enter STVATYP in the Entity field.		Entities should be named after the table for which the crosswalk is needed.
Perform a NEXT BLOCK function.		Cursor is in the Legacy Value field.
Enter the two values from the data in the Legacy Value fields (L,P), and enter their new SCT Banner equivalents in the appropriate SCT Banner value fields (MA, PR).		For this exercise, L = MA, P = PR. Please verify that these values exist in your training database. During your conversion, you may need to research the values that are being used by your functional users and get their assistance in creating crosswalks.
SAVE and EXIT the Crosswalk form.		
Navigate to the Convert Function field and click the CONVERT FUNCTION button. SELECT F_CVT_CURCVAL_RNULL.		
With your cursor in the Convert Function field, click the View Function button. When the Function Code and Parameters window appears, click the EDIT PARAMETERS button.		
Enter "STVATYP" as the entity.		Must be enclosed in single quotes.
Enter LEGACY_VALUE in the LEGACY_VALUE field.		No quotes.
Click the CONSTRUCT FUNCTION CALL button.		
Click RETURN, then SAVE and RETURN.		You will need to perform a NEXT BLOCK function to move to the column block of the form.
Navigate to the Load and Insert fields. Check both.		
Navigate to the Next Record, SPRADDR_SEQNO		
Navigate to the Convert Function field and click the CONVERT FUNCTION button. SELECT F_GEN_SEQUENCE.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
With your cursor in the Convert Function field, click the View Function icon on the toolbar.		
Click the Edit Parameters button.		
Enter SEQ1 as the parameter for SEQUENCE.		
Click the CONSTRUCT FUNCTION CALL button.		
Click RETURN/SAVE & RETURN. The cursor will be in the Owner field of the Table Block..		
Click the BREAKPOINT INFO button in the Table Block. Enter the following data: BREAKPOINT 1 = CONVERT_PIDM BREAK FUNCTION = F_GEN_RESET_SEQNO(SEQ1) BREAKPOINT 2 = CONVERT_ATYP_CODE BREAK FUNCTION = F_GEN_RESET_SEQNO(SEQ1)		You may choose to click the BREAK FUNCTION button. It will bring up the function library list. From there you may choose the F_GEN_RESET_SEQNO function and populate the parameters from the View Function window. OR you may simply enter the data as shown. This function will apply a set of sequence numbers for every series of pidm/atyp_codes that exist. On a change in that combination of fields, the sequence number is reset and the series begins again with a new pidm/atyp_code set.
Close the window by clicking the X in the upper right corner. SAVE your changes.		"Breakpoint Exists" message appears on form beside BREAKPOINT INFO button.
Navigate to the Column Block, with the cursor in the SPRADDR_SEQNO column.		
Navigate to the Load and Insert boxes. UNcheck Load, check Insert.		There is no data to load, but we must insert the generated sequence number.
Navigate to the next record, SPRADDR_FROM_DATE. Navigate to the Format Mask field and enter YYYYMMDD.		We have data for a From Date on two of the sample records.
Navigate to the Load and Insert boxes. Check both.		
Navigate to the next record, SPRADDR_TO_DATE. Navigate to the Format Mask field and enter YYYYMMDD.		We have data for a To Date on two of the sample records.
Navigate to the Load and Insert boxes. Check both.		
Navigate to the SPRADDR_STREET_LINE1 column. Navigate to the Load & Insert boxes. Check both.		For our exercise, we are not adjusting the size of the convert columns for the address. You may choose to leave the size as it is, in order to identify any records that will not meet SCT Banner's size constraints for addresses. OR you may choose to adjust the size and discover the incompatible records during the Insert phase.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the SPRADDR_STREET_LINE2 column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_STREET_LINE3 column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_CITY column. Navigate to the Load & Insert boxes. Check both.		
Navigate to the SPRADDR_STAT_CODE column. Navigate to the Load & Insert boxes. Check both.		For this exercise, we will assume that there is no need for a crosswalk because we expect the SEED data in your training database to contain standard state codes.
Navigate to the SPRADDR_ZIP column. Navigate to the Load & Insert boxes. Check both.		
Navigate to the SPRADDR_CNTY_CODE column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_NATN_CODE column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_PHONE_AREA column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_PHONE_NUMBER column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_PHONE_EXT column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_STATUS_IND column. Navigate to the Load & Insert boxes. UNcheck both.		
Navigate to the SPRADDR_ACTIVITY_DATE column. Navigate to the Convert Function field. Enter SYSDATE.		
Navigate to the Load & Insert boxes. UNcheck Load. Check Insert.		
Navigate through all remaining columns. UNcheck both Load and Insert.		
SAVE your changes.		

You have now entered all the necessary specifications for the SPRADDR table conversion. The next step is to use the features of the converter tool to produce the following 3 scripts:

- spraddr_cvt_create.sql – creates the convert table (SPRADDR_CVT). To create this script, press the CONVERT TABLE SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script should be run in SQL on the server to create the convert table in the host database.
- spraddr_cvt.ctl – used by SQL*Loader to load the data from the client's flat file (spraddr_cvt.dat) to the convert table (SPRADDR_CVT). To create this script, press the SQL LOADER SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script must be placed in the same directory on the server with the data file (spraddr_cvt.dat). Run SQL*Loader on the server to load the data from the flat file to the convert table *convert* columns.
- spraddr_convert.sql – moves the data from the *convert* columns to the target table columns within the convert table, then moves the data from the target table columns in the convert table to the corresponding columns in the actual target table in the database. To create this script, press the CONVERSION SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script should be run in SQL on the server.

Establishing Specifications for Conversion of SPBPERS Table

The following steps will show how to use the SCT Converter Tool to set up specifications for converting data into the SCT Banner SPBPERS table:

Note: This exercise will demonstrate the use of the Load Order feature and the F_VALIDATE_SINGLE function.

Sample legacy data for this file includes the following values, in this order. *pidm, ssn, birth date, marital code (values Y, M, D, W, S), ethn code (values 1,2,3,M), sex (values F, M, N)*

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Open CUACNVT and log in. Login: your_converter_username Password: your_convert_user_password Database: your_database	CUACNVT	
Insert a new record	CUACNVT Table Block	Record/Insert on the Menu OR Press the F6 key OR Use the down arrow key to Navigate to a blank record.
Enter the name of the table owner, SATURN.		
Navigate to the Next Record.		
Enter the name of the table, SPBPERS.		
Navigate to the “Error Action” field.		Accept the default Error Action, “Continue Processing Row.”
SAVE your changes		
NEXT BLOCK	CUACNVT Column Block	
Cursor is in the SPBPERS_PIDM field.		
Navigate to the “Convert Fnctn” field.		
Press the CONVERT FNCTN button.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
SELECT "F_CVT_GET_PIDM" from the list.		The assumption is that the legacy data has already been loaded into the SPRIDEN table. This function uses the legacy ID to match with SPRIDEN_ID. If a match is found, the existing SPRIDEN_PIDM is returned to populate the SPBPERS_PIDM field. If no match is found, the transaction will produce an error.
With your cursor in the "Convert Fnctn" field, press the VIEW FUNCTION button on the Tool Bar.	Function Code and Parameters Window	
Click the EDIT PARAMETERS button.	Edit Parameters Dialog Box	The parameter for the F_CVT_GET_PIDM function is ID.
Enter LEGACY_VALUE in the right column of the parameters dialog box.		In the spbpers_cvt table, the legacy value that will be loaded into the *convert_pidm* column is the legacy ID. That is the value that must be passed to the function. In the "Length" field, we will change the number from 8 to 9 to accommodate the legacy ID value that will be loaded into the convert table. This action will cause the *convert_pidm* column in the convert table to have a length of 9, while the *spbpers_pidm* column in the convert table retains its prescribed length of 8.
Press the CONSTRUCT FUNCTION CALL button.		When you press the CONSTRUCT FUNCTION CALL button, the blank box at the top right corner of the form is populated with the actual function call that will appear in the convert script: F_CVT_GET_PIDM(LEGACY_VALUE)
Press the RETURN button.		
Choose "Save & Return."		Doing this will return your cursor to the Table Block of CUACNVT. To resume work, you must perform a NEXT BLOCK function.
NEXT BLOCK	CUACNVT	
With your cursor in the SPBPERS_PIDM column field, Navigate to the "Length" field.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Change the 8 to 9 (no quotation marks)		You must change the length of the CONVERT_PIDM column in the table to accommodate the legacy ID, which is 9 characters long. The F_CVT_GET_PIDM function will use this ID to match against the existing SPRIDEN_ID, inserted previously, to get the pidm and insert it into the SPBPERS_PIDM field in the convert table and subsequently into the SPBPERS_PIDM field in the SPBPERS table.
Navigate to the "Load" field in the Column Block and check the "Load" indicator.		Checking the LOAD button indicates that you want this column to appear in the SQL*Loader script because you have legacy data in the flat file for this column to be loaded into the convert table.
Navigate to the "Insert" indicator in the Column Block and check the "Insert" indicator.		Checking the INSERT button indicates that you want this column to appear in the conversion script to be converted to the target table columns in the convert table and subsequently inserted into the actual target table.
Navigate to the next record, the SPBPERS_SSN column. Verify that the Load and Insert boxes are checked	CUACNVT Column Block	
Navigate to the SPBPERS_BIRTH_DATE column.		
Navigate to the Format Mask field. Enter YYYYMMDD		
Verify that the Load and Insert boxes are checked.		
Navigate to the SPBPERS_LGCY_CODE column. Navigate to the Load and Insert boxes. UNcheck both.		
Navigate to the SPBPERS_ETHN_CODE column. Navigate to the Load Order field. Change the number from "5" to "6."		The Load Order feature allows you to establish the order of the columns in the SQL*Loader control file. This could be helpful if for some reason the data elements in your data file were in an order different from that required by the SCT Banner table. In this exercise, our data file places the ethnic code legacy value ahead of the marital code legacy value, forcing us to reverse the load order for these two fields. Changing the load order does not affect the order ot the columns in the Insert phase into the SCT Banner tables, nor does it affect the architecture of the convert table. NOTE: Once you save your changes and re-enter the record for SPBPERS, the columns will have been re-ordered the way you have specified.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Click the Crosswalk icon to display the Crosswalk form (CUACVAL)		
Enter STVETHN in the Entity field.		Entities should be named after the table for which the crosswalk is needed.
Perform a NEXT BLOCK function.		Cursor is in the Legacy Value field.
Enter the values from the data in the Legacy Value fields (1,2,3,M), and enter their new SCT Banner equivalents in the appropriate SCT Banner value fields .		For this exercise, you will need to discover the ethnic codes in the training database where you are working and create some equivalents. During your conversion, you may need to research the values that are being used by your functional users and get their assistance in creating crosswalks.
SAVE and EXIT the Crosswalk form.		
Navigate to the Convert Function field and click the CONVERT FUNCTION button. SELECT F_CVT_CURCVAL_RNULL.		
With your cursor in the Convert Function field, click the View Function button. When the Function Code and Parameters window appears, click the EDIT PARAMETERS button.		
Enter “STVETHN” as the entity.		Must be enclosed in single quotes.
Enter LEGACY_VALUE in the LEGACY_VALUE field.		No quotes.
Click the CONSTRUCT FUNCTION CALL button.		
Click RETURN, then SAVE and RETURN.		You will need to perform a NEXT BLOCK function to move to the column block of the form.
Navigate to the Load and Insert fields. Check both.		
Navigate to the SPBPERS_MRTL_CODE column.		
Navigate to the Load Order field and change the “6” to “5.”		
Navigate to the Validate Function field and click the VALID FUNCTION button.		
SELECT F_VALIDATE_SINGLE from the list.		
With your cursor in the Validate Function field, click the View Function icon on the Toolbar.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Click the EDIT PARAMETERS button and enter the following parameters: OWNER = 'SATURN' TABLE = 'STVMRTL' COLUMN = 'STVMRTL_CODE' IN_VALUE = LEGACY VALUE		Note: OWNER, TABLE, and COLUMN parameters must be enclosed in single quotes. IN_VALUE should not be enclosed in quotes.
Click the CONSTRUCT FUNCTION CALL button. Click RETURN/SAVE & RETURN. When the CUACNVT form displays, perform a NEXT BLOCK function.		
Verify that the Load and Insert boxes are checked for the SPBPERS_MRTL_CODE column.		
Navigate to the SPBPERS_RELG_CODE column.		
Navigate to the Load and Insert boxes. UNcheck both.		There is no data for this field.
Navigate to the SPBPERS_RELG_CODE column. Navigate to the Load and Insert boxes and UNcheck both.		There is no data for this field.
Navigate to the SPBPERS_SEX column.		
Navigate to the Value List field and insert the following, in this format: "M", "F", "N"		Any record having a legacy value other than one in the list will create an error.
Naviegate to the SPBPERS_CONFID_IND. UNcheck the Load and Insert boxes. Navigate through each of the following columns, UNchecking the Load and Insert boxes on each one: SPBPERS_CONFID_IND SPBPERS_DEAD_IND SPBPERS_VETC_FILE_NUMBER SPBPERS_LEGAL_NAME SPBPERS_PREF_FIRST_NAME SPBPERS_NAME_PREFIX SPBPERS_NAME_SUFFIX		There is no data for any of these fields.
Navigate to the SPBPERS_ACTIVITY_DATE column. Navigate to the Convert Function field and enter SYSDATE.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the Load and Inser boxes. UNcheck the Load box, CHECK the Insert box.		
Navigate through ALL remaining columns in SPBPERS, UNchecking both the Load and Insert boxes for each column.		
SAVE your changes.		

You have now entered all the necessary specifications for the SPBPERS table conversion. The next step is to use the features of the converter tool to produce the following 3 scripts:

- SPBPERS_cvt_create.sql – creates the convert table (SPBPERS_CVT). To create this script, press the CONVERT TABLE SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script should be run in SQL on the server to create the convert table in the host database.
- SPBPERS_cvt.ctl – used by SQL*Loader to load the data from the client's flat file (SPBPERS_cvt.dat) to the convert table (SPBPERS_CVT). To create this script, press the SQL LOADER SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script must be placed in the same directory on the server with the data file (SPBPERS_cvt.dat). Run SQL*Loader on the server to load the data from the flat file to the convert table *convert* columns.
- SPBPERS_convert.sql – moves the data from the *convert* columns to the target table columns within the convert table, then moves the data from the target table columns in the convert table to the corresponding columns in the actual target table in the database. To create this script, press the CONVERSION SCRIPT button on the Tool Bar. You will see the success message on the message line of the form. This script should be run in SQL on the server.

Chapter 4 - Scripts Produced by the SCT Converter Tool (CAUCNVT)

Create Convert Table Script: <*TargetTable*>_*cvt_create.sql* 4-3
Create SQL*Loader Script: <*TargetTable*>_cvt.ctl 4-5
Create Convert Script: <*TargetTable*>_convert.sql 4-7
Running the Convert Script 4-8
Answering the Prompts 4-8
Explanation of Record Type Prompts 4-8
Dealing with Errors 4-11

Scripts Produced by the SCT Converter Tool (CAUCNVT)

Create Convert Table Script: <*TargetTable*>_cvt_create.sql

After building converstion specifications on the CUACNVT form, the next step is to create an Oracle table to hold the data from the flat file the client has provided.

The SCT Converter Tool enables you to produce a script (*<TargetTable>_cvt_create.sql)* that creates a convert table which contains each target table column as well as a *convert* column for each of the columns in the target table. The *convert* columns are exact replicas of the target table columns in size and position, but without the constraints that exist on the target table columns. *Convert* columns are always the VARCHAR2 datatype.

In one of the sample conversions outlined in this document, the SPRIDEN table (Person Identification Table) is being loaded. The SPRIDEN table includes the following columns:

Column	Data Type	Constraints
SPRIDEN_PIDM	Number (8)	Not Null
SPRIDEN_ID	Varchar2 (9)	Not Null
SPRIDEN_LAST_NAME	Varchar2 (60)	Not Null
SPRIDEN_FIRST_NAME	Varchar2 (15)	
SPRIDEN_MI	Varchar2 (15)	
SPRIDEN_CHANGE_IND	Varchar2 (1)	
SPRIDEN_ENTITY_IND	Varchar2 (1)	
SPRIDEN_ACTIVITY_DATE	Date	Not Null
SPRIDEN_USER	Varchar2 (30)	
SPRIDEN_ORIGIN	Varchar2 (30)	
SPRIDEN_SEARCH_LAST_NAME	Varchar2 (60)	
SPRIDEN_SEARCH_FIRST_NAME	Varchar2 (15)	
SPRIDEN_SEARCH_MI	Varchar2 (15)	
SPRIDEN_SOUNDEX_LAST_NAME	Char (4)	
SPRIDEN_SOUNDEX_FIRST_NAME	Char (4)	
SPRIDEN_NTYP_CODE	Varchar2 (4)	

The flat file that we are using in our example contains the legacy data for the SPRIDEN_ID, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, and SPRIDEN_MI columns (to see the contents of the flat files used in the example, refer to Appendix A: Sample Data Files Used in Examples). Other required and/or desired fields for the convert table are SPRIDEN_PIDM, SPRIDEN_ENTITY_IND, SPRIDEN_ACTIVITY_DATE, SPRIDEN_USER, and SPRIDEN_ORIGIN.

All other fields are excluded from the Load and/or Insert process on the CUACNVT form by checking or unchecking the Load and Insert buttons. The *spriden_cvt_create.sql* script, however, contains all columns in the SPRIDEN target table plus a set of convert columns that mirror the target table columns. The differences between the columns are 1) the convert columns are all VARCHAR2 data types, and 2) the size of the convert column may vary from the size of the target table column if users have specified changes on the CUACNVT form during setup.

This is a sample of a table creation script, *spriden_cvt_create.sql* generated by the converter tool.

```
/*
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   09/30/2002
  This script creates the temporary conversion table SPRIDEN for
  converting legacy data into SPRIDEN.  This script was generated
  by the SCT Converter Tool.

AUDIT TRAIL END
*/
DROP TABLE SPRIDEN_CVT;
CREATE TABLE SPRIDEN_CVT
(
SPRIDEN_PIDM NUMBER(8),
CONVERT_PIDM VARCHAR2(9),
SPRIDEN_ID VARCHAR2(9),
CONVERT_ID VARCHAR2(9),
SPRIDEN_LAST_NAME VARCHAR2(60),
CONVERT_LAST_NAME VARCHAR2(60),
SPRIDEN_FIRST_NAME VARCHAR2(15),
CONVERT_FIRST_NAME VARCHAR2(15),
SPRIDEN_MI VARCHAR2(15),
CONVERT_MI VARCHAR2(15),
SPRIDEN_CHANGE_IND VARCHAR2(1),
CONVERT_CHANGE_IND VARCHAR2(1),
SPRIDEN_ENTITY_IND VARCHAR2(1),
CONVERT_ENTITY_IND VARCHAR2(1),
SPRIDEN_ACTIVITY_DATE DATE,
CONVERT_ACTIVITY_DATE VARCHAR2(9),
SPRIDEN_USER VARCHAR2(30),
CONVERT_USER VARCHAR2(30),
SPRIDEN_ORIGIN VARCHAR2(30),
CONVERT_ORIGIN VARCHAR2(30),
SPRIDEN_SEARCH_LAST_NAME VARCHAR2(60),
CONVERT_SEARCH_LAST_NAME VARCHAR2(60),
SPRIDEN_SEARCH_FIRST_NAME VARCHAR2(15),
CONVERT_SEARCH_FIRST_NAME VARCHAR2(15),
SPRIDEN_SEARCH_MI VARCHAR2(15),
CONVERT_SEARCH_MI VARCHAR2(15),
SPRIDEN_SOUNDEX_LAST_NAME CHAR(4),
CONVERT_SOUNDEX_LAST_NAME VARCHAR2(4),
SPRIDEN_SOUNDEX_FIRST_NAME CHAR(4),
CONVERT_SOUNDEX_FIRST_NAME VARCHAR2(4),
SPRIDEN_NTYP_CODE VARCHAR2(4),
CONVERT_NTYP_CODE VARCHAR2(4),
SPRIDEN_CVT_RECORD_ID NUMBER(8) CONSTRAINT PK_SPRIDEN_CVT
PRIMARY KEY,
SPRIDEN_CVT_STATUS VARCHAR2(1),
SPRIDEN_CVT_JOB_ID NUMBER(8) )
STORAGE (INITIAL 1M
      PCTINCREASE 100
      MAXEXTENTS 50);
```

The convert column after each SPRIDEN column uses the naming convention that substitutes the word *"convert"* for the SPRIDEN table name.

Also, three new columns have been added:

- The <*TargetTable*>*_cvt_record_id* column holds a sequence number which is assigned when the table is loaded using the SQL*Loader control file produced by the Converter Tool.
- The <*TargetTable*>*_cvt_status* column holds the record status (New, Converted, or Error) assigned during the conversion process (Explained in more detail in the section on the convert script).
- The <*TargetTable*>*_cvt_job_id* column holds a unique number which is used in naming the log files that are created for each run of the convert script, allowing each log file to remain intact for reference.

This script should be run in SQL on the client machine in the correct database instance to create the temporary Oracle table.

Caution! The script includes a DROP TABLE statement. The table (including data) is removed. Run this script only if the data in the convert table in not needed.

Create SQL*Loader Script: <*TargetTable*>_cvt.ctl

The second script created by the converter tool is the loader script, or control file that is used by the SQL*Loader utility to populate the <T*argetTable*>*_cvt* convert table with the flat file data.

The control file that is produced will contain a listing of the convert columns that parallel the <*TargetTable*> columns selected on the CUACNVT form by checking the "Load" indicator, as well as the <*TargetTable*>*_convert_record_id* and the <*TargetTable*>*_cvt_status* columns.

Note: If you have data in the flat file that you want to load into the convert table, you *must* check the "Load" indicator on the CUACNVT form.

In our sample conversion, this is the flat file that is loaded by the loader process:

File Name: spriden_cvt.dat
Legacy Fields: legacy ID, legacy ID, last name, first name, middle name,change_ind, entity_ind
SCT Banner Fields: spriden_id, spriden_id, spriden_last_name, spriden_first_name, spriden_mi, spriden_change_ind, spriden_entity_ind

DATA:
```
                ,111111111,SMITH,AMELIA,KAY,,P
111111111,101101101,SMITH,AMELIA,KAY,I,P
                ,222222222,SIMONEAUX,MARYANNE,FRANCIS,,P
222222222,222222222,CAMPESI,MARYANNE,S,N,P
                ,333333333,WOODRUFF,LUCILLE,MARIE,,P
                ,444444444,CAMPBELL,JOSEPH,ALEXANDER,,P
                ,555555555,PITTMANN,MILDRED,YATES,,P
                ,666666666,BOYD,WILLIAM,GUERRY,,P
                ,777777777,GLOVER,SAVION,TAP,,P
                ,888888888,WINGFIELD,WILLIAM,DOUGLASS,,P
                ,999999999,LENNON,JOHN,O,,P
                ,000000000,ELDRIDGE,GROVER,BLUE,,P
```

This is a sample control file script, *spriden_cvt.ctl,* generated by the converter tool for the SPRIDEN table:

```
-- AUDIT TRAIL
-- Converter Tool:  2.10                    SCTCVT   09/30/2002
--    This SQLLoader control file is used to load legacy data into
--    the temporary conversion table SPRIDEN_cvt.  This script
--    was generated by the SCT Converter Tool.
--
-- AUDIT TRAIL END
Load data
Infile 'spriden_cvt.dat'
APPEND
INTO TABLE SPRIDEN_CVT
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' TRAILING
NULLCOLS
(
CONVERT_PIDM CHAR,
CONVERT_ID CHAR,
CONVERT_LAST_NAME CHAR,
CONVERT_FIRST_NAME CHAR,
CONVERT_MI CHAR,
CONVERT_CHANGE_IND CHAR,
SPRIDEN_CVT_RECORD_ID SEQUENCE(MAX,1),
SPRIDEN_CVT_STATUS CONSTANT 'N'
)
```

The control file includes convert columns for the data fields specified on CUACNVT by the "Load" indicator, in the correct order. Note the loading of default values into the SPRIDEN_CVT_RECORD_ID column and the SPRIDEN_CVT_STATUS columns.

Notice the "Infile" name. The flat file must be re-named to conform to this naming convention: <*TargetTable*>*_cvt.dat*

The <*TargetTable*>_CVT_RECORD_ID_SEQUENCE and the <*TargetTable*>_CVT_STATUS columns also appear in the control file. Values will be inserted in the load process.

The <*TargetTable*>_CVT_RECORD_ID_SEQUENCE will be populated with a sequence number to identify the record, and the <*TargetTable*>_CVT_STATUS will be populated with the value "N" to indicate that the record is (N)ew. This value will be updated when the convert script is run in the later stages of the process. Other valid values for this field during the convert process are (C)onverted, (I)nserted and (E)rrored.

Notice also that the columns that are to be loaded are only the *convert* columns, and not the <*TargetTable*> columns. <*TargetTable*> columns are populated in the final step, using the conversion script.

When you create the specifications for your conversion of the <*TargetTable*>, the SCT Converter Tool allows you to specify the format of your flat file (comma delimited or sequential/fixed length) and will create the appropriate type of control file, depending upon your selection.

After generating the control file from the converter tool, you will need to place the data file and the control file in the same directory on the server to load the convert table you have created. Once the two files are in the same directory on the server, use the following command to invoke SQL*Loader:

sqlldr userid=your_userid/your_convert_user_password control=<*TargetTable*>*_cvt.ctl*

Note: The command to invoke SQL*Loader (sqlldr) may vary from platform to platform. Please check with your DBA to get the correct command.

Create Convert Script: <*TargetTable*>_convert.sql

The third script produced by the converter tool is the "*convert"* script. This script is run on the server after the legacy data from the flat file has been loaded into the *convert* columns of the convert table.

At this point, you have included any functions, variables, decodes, or default values on CUACNVT necessary to manipulate the data so that it is compatible with the <*TargetTable*> constraints. Additionally, all appropriate columns to be inserted have been coded as such with the "Insert" indicator on the form.

The convert script, <*TargetTable*>*_convert.sql,* generated by the converter tool, incorporates all the specifications set up on the CUACNVT form and applies them as the convert script is run.

The convert script can be run in "Convert" mode or "Insert" mode. Running it in "Convert" mode copies the legacy data from the *convert* columns of the convert table to the <*TargetTable*> columns of the convert table, performing any specified data transformation during processing. Running the script in "Insert" mode inserts the values from the <*TargetTable*> columns in the convert table into the actual <*TargetTable*> columns of the target table.

There is the option for the convert script to both "Convert" and "Insert" in the same transaction. This is convenient for loading small, simple tables such as validation tables, where there is a strong confidence in the accuracy and compatibility of the data with the <*TargetTable*>. For most conversions, however, it is recommended that the script be run in "Convert" mode until all errors are corrected.

Specifying Rollback Segments for the Conversion Process:

Depending upon your version of Oracle, for large conversion files, you may need to ask your DBA to create a large rollback segment (typically LRG_RBS1) if one does not exist already. Before running the conversion script against a large convert table, you will need to take the smaller rollback segments off-line and be sure that the large rollback segment is online.

To do this, ask your DBA to execute the following commands at the SQL> prompt: [note that these rollback segment names are samples – your environment may differ]:

```
alter rollback segment RBS1 offline

/

alter rollback segment RBS2 offline

/

alter rollback segment RBS3 offline

/

alter rollback segment LRG_RBS1 online

/
```

Running the Convert Script

Answering the Prompts

First set of Prompts: Record Type

When starting the convert script, you are prompted for a "record type" and given the following 3 choices:

Please enter the record type which should be processed.

(N)ew records

(C)onverted records

(E)rrored records

These values are stored in the <*TargetTable*>*_cvt_status* field in the convert table (<*TargetTable*>*_cvt).* This column holds the indicators for the status of the record. Valid values are "N"=New; "C"=Converted; "I" = Inserted, "E"=Error.

All records are assigned a default value of "N" (New) when the convert table is loaded using the <TargetTable>_cvt.ctl loader script. The value is changed to "C" when the record is successfully copied to the <*TargetTable*> column in the convert table through the convert process. The value is changed to "E" if the record cannot be copied to the <*TargetTable*> column in the convert table because of a constraint or other error.

Explanation of Record Type Prompts:

(N) If you are running the convert script for the first time, you should choose (N)ew. When data is loaded into the convert table for the first time, all records are marked with an indicator of "N" for "New".

Choosing (N) at this prompt instructs the script to use the values in the convert columns, apply any special functions or instructions that you indicated on CUACNVT when you created the script. It then copies those values into the <*target_table*> columns in the convert table.

(C) The second choice at the first prompt, (C)onverted, will instruct the script to select only those records marked with "C". Records are marked with a "C" once the script has been run in "Convert" mode at least once (explained below) and these records have been successfully inserted into the <*target_table*> columns of the convert table. You would choose (C) if you have successfully converted the data and are ready to insert it into the actual <*TargetTable*> columns.

(E) Any records that have errors are not loaded into the <*TargetTable*> columns and are marked with an "E" in the <*TargetTable*>*_cvt_status* column, so they are easily identifiable for evaluation and correction.

After you have corrected any records that errored out on the first run of the script, you can run the script again, and answer the first prompt with (E), indicating that you wish only to process the records that were previously marked as errors on the first run. You may repeat this process as many times as is necessary to correct all errors.

Second set of Prompts: Convert Disposition

Please enter the disposition to which the records should be processed.

(C)onvert records only

(I)nsert records only (records were previously converted successfully).

(B)oth convert and insert records into <TargetTable>

(C) Convert records only: Answer with "C" if you wish the script only to populate the SCT Banner columns in the convert table and not to insert values into the target SCT Banner table. This will allow you to evaluate and correct any errored records, while not touching the actual SCT Banner tables.

(I) Insert records that were previously converted successfully. Choose this option after you have run the script at least once in Convert mode to check for errors. Choosing (I) at this prompt instructs the script to insert converted values into the target SCT Banner table in the database.

(B) Both Convert and Insert records into actual SCT Banner table. Choosing (B) at this prompt will instruct the script to move the values from the convert columns in the convert table to the <TargetTable> columns in the convert table, and then to insert the values from the <TargetTable> columns in the convert table into the actual <TargetTable>. All of these transactions are performed in just one run of the convert process. Errored records will remain in the convert table, marked with <TargetTable>_cvt_status of "E".

When the script completes its run, you will see the following output:

- Number of Rows Converted in <TargetTable>_CVT:
- Number of Rows Inserted into <TargetTable>:
- Number of Rows That Errored:
- Completed Processing of <TargetTable>;

You are told how many rows were converted into the <TargetTable> columns in the convert table, how many rows were inserted into the actual <TargetTable>, and how many rows had errors that prevented them from being loaded.

How the Convert Script Processes the Specifications Set up on CUACNVT:

Here are two excerpts from the spriden_convert.sql script. Notice the following items, highlighted in the script:

- function call in the SPRIDEN_PIDM section with the passed parameter of "LEGACY_VALUE"
- the Default value 'SCTCVT' in the SPRIDEN_USER section

These values were entered in the CUACNVT form.

```
PROMPT ================================================
PROMPT = will be creating generated id records       =
PROMPT = in SPRIDEN_CVT for any converted records    =
PROMPT =  (i.e. records with spriden_cvt_status = 'C' =
PROMPT ================================================

/*-------------------------------
    |Beginning evaluation of column
    |SPRIDEN_PIDM
    --------------------------------*/
    BEGIN
      cur_col := 'SPRIDEN_PIDM';
      legacy_value := spriden_rec.convert_pidm;
      /* CURCNVT Rules:
        Column is Required
        Conversion function is specified
        No validation specified
        No default value
      */
```

```
      converted_val := null;
      converted_val := F_CVT_AUTO_PIDM(LEGACY_VALUE);
      IF SUBSTR(converted_val,1,3) = 'ERR' OR
        SUBSTR(converted_val,1,3) = 'ORA' THEN
        err_msg := 'Column conv failure:'||converted_val;
        RAISE column_error;
      END IF;
          ….

    /*-------------------------------
      |Beginning evaluation of column
      |SPRIDEN_USER
      ---------------------------------*/
      BEGIN
        cur_col := 'SPRIDEN_USER';
        legacy_value := spriden_rec.convert_user;
        /* CURCNVT Rules:
          Column is Not Required
          Conversion function is not specified
          No validation specified
          Default value 'SCTCVT'
          Default Action 'D' - Default if null legacy value
        */
        converted_val := null;
        converted_val := NVL(spriden_rec.CONVERT_USER,'SCTCVT');
        spriden_rec.SPRIDEN_USER := converted_val;
      EXCEPTION
        WHEN column_error THEN
          col_status := 'E';
          process_status := 'E';
          p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
          legacy_value,converted_val,null,err_msg);
      END;

  /*-------------------------------
      |Beginning evaluation of column
      |SPRIDEN_ENTITY_IND
      ---------------------------------*/
      BEGIN
        cur_col := 'SPRIDEN_ENTITY_IND';
        legacy_value := spriden_rec.convert_entity_ind;
        /* CURCNVT Rules:
          Column is Not Required
          Conversion function is not specified
          No validation specified
          Default value 'P'
          Default Action 'D' - Default if null legacy value
        */
        converted_val := null;
        converted_val := NVL(spriden_rec.CONVERT_ENTITY_IND,'P');
        spriden_rec.SPRIDEN_ENTITY_IND := converted_val;
```

Dealing with Errors

If your data has errors, the message after the convert script is run will indicate the number of errors that have occurred.

- Number of Rows Converted in <*TargetTable*>_CVT:
- Number of Rows Inserted into <*TargetTable*>:
- Number of Rows That Errored:

To view the details about the error, go to the CUACNVT form, click the VIEW ERRORS button, enter the table name in the key block, and perform a Next Block function. You will see the record detail, the Oracle error message and more.

You may also get error information from the CURCERR table while at the SQL> prompt. This is the structure of the CURCERR table:

Column	Data Type	Constraints
CURCERR_TABLE_NAME	Varchar2 (30)	
CURCERR_CVT_IDENTIFIER	Number (3)	
CURCERR_RECORD_ID	Number (8)	
CURCERR_COLUMN_NAME	Varchar2 (30)	
CURCERR_LEGACY_VALUE	Varchar2 (100)	
CURCERR_BANNER_VALUE	Varchar2 (100)	
CURCERR_ERRNO	Number (6)	
CURCERR_MESSAGE	Varchar2 (300)	
CURCERR_ACTIVITY_DATE	Date	
CURCERR_TABLE_OWNER	Varchar2 (30)	


sct

Appendix A - Sample Data Files Used in Examples

ASCII Data Files Used in Conversion Examples: A-3
spriden_cvt.dat* A-3
spraddr_cvt.dat A-3
spbpers_cvt.dat A-4

ASCII Data Files Used in Conversion Examples:

spriden_cvt.dat*

Legacy Fields: legacy ID, legacy_ID,last name, first name, middle name,change indicator, entity indicator

SCT Banner Fields: spriden_pidm, spriden_id, spriden_last_name, spriden_first_name, spriden_mi, spriden_change_ind, spriden_entity_ind

DATA: ,111111111,SMITH,AMELIA,KAY,,P
111111111,101101101,SMITH,AMELIA,KAY,I,P
,222222222,SIMONEAUX,MARYANNE,FRANCIS,,P
222222222,222222222,CAMPESI,MARYANNE,S,N,P
,333333333,WOODRUFF,LUCILLE,MARIE,,P
,444444444,CAMPBELL,JOSEPH,ALEXANDER,,P
,555555555,PITTMANN,MILDRED,YATES,,P
,666666666,BOYD,WILLIAM,GUERRY,,P
,777777777,GLOVER,SAVION,TAP,,P
,888888888,WINGFIELD,WILLIAM,DOUGLASS,,P
,999999999,LENNON,JOHN,O,,P
,000000000,ELDRIDGE,GROVER,BLUE,,P

Note: Please refer to the "Loading Data into the SPRIDEN Table" section of this document for a detailed explanation of the file layout for the spriden table.

spraddr_cvt.dat

Legacy Fields: legacy ID, address type, begin date, end date, street, city, state, zip

SCT Banner Fields: spraddr_pidm, spraddr_atyp_code, spraddr_from_date, spraddr_to_date, spraddr_street_line1, spraddr_city, spraddr_stat_code, spraddr_zip

DATA: 111111111,L,20010101,20011212,3918 Wheat Street,Columbia,SC,29205
111111111,L,20020101,,831 Timberlane Drive, Rochester, NY,32154
222222222,P,20010101,20011212,222 Sun Valley Drive,Americus,GA,39102
222222222,P,20010101,,999 Tappenzee Street, New Amsterdam,NY,98765
333333333,,,,333 Tugaloo Ave.,Columbia,SC,29205
444444444,,,,444 Fairway Oaks Drive,Brunswick,GA,31520
555555555,,,,555 Broadway,New York,NY,21222
666666666,,,,666 Rodeo Drive,Los Angeles,CA,66666
777777777,,,,777 Lucky Lady Lane,Cherokee,NC,77777
888888888,,,,888 Crazy Eight Road,Hoyle,WV,88888
999999999,,,,999 John Lennon Way,Revolution City,ND,99999
000000000,,,,000 Zero Street,Nowhere,NJ,00000

spbpers_cvt.dat

Legacy Fields: legacy ID, legacy SSN, birth date, marital code, ethnic code, sex
SCT Banner Fields: spbpers_pidm, spbpers_ssn, spbpers_birth_date, spbpers_mrtl_code, spbpers_ethn_code, spbpers_sex

DATA:
```
111111111,111111111,01017077,Y,2,F
222222222,222222222,19480203,M,2,F
333333333,333333333,19750412,D,1,T
444444444,444444444,19770309,W,2,M
555555555,555555555,19780815,S,1,F
666666666,666666666,19480128,Y,2,M
777777777,777777777,19552323,M,M,M
888888888,888888888,19500406,D,3,F
999999999,999999999,19630106,W,2,M
000000000,000000000,19540321,S,1,N
```

Appendix B - Sample Table Creation Scripts

Sample <TargetTable>_cvt_create.sql scripts B-3
spriden_cvt_create.sql B-3
spraddr_cvt_create.sql B-4
spbpers_cvt_create.sql B-5

Sample <TargetTable>_cvt_create.sql scripts

spriden_cvt_create.sql

```
/*
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT      09/30/2002
  This script creates the temporary conversion table SPRIDEN for
  converting legacy data into SPRIDEN.  This script was generated
  by the SCT Converter Tool.

AUDIT TRAIL END
*/
DROP TABLE SPRIDEN_CVT;
CREATE TABLE SPRIDEN_CVT
(
SPRIDEN_PIDM NUMBER(8),
CONVERT_PIDM VARCHAR2(9),
SPRIDEN_ID VARCHAR2(9),
CONVERT_ID VARCHAR2(9),
SPRIDEN_LAST_NAME VARCHAR2(60),
CONVERT_LAST_NAME VARCHAR2(60),
SPRIDEN_FIRST_NAME VARCHAR2(15),
CONVERT_FIRST_NAME VARCHAR2(15),
SPRIDEN_MI VARCHAR2(15),
CONVERT_MI VARCHAR2(15),
SPRIDEN_CHANGE_IND VARCHAR2(1),
CONVERT_CHANGE_IND VARCHAR2(1),
SPRIDEN_ENTITY_IND VARCHAR2(1),
CONVERT_ENTITY_IND VARCHAR2(1),
SPRIDEN_ACTIVITY_DATE DATE,
CONVERT_ACTIVITY_DATE VARCHAR2(9),
SPRIDEN_USER VARCHAR2(30),
CONVERT_USER VARCHAR2(30),
SPRIDEN_ORIGIN VARCHAR2(30),
CONVERT_ORIGIN VARCHAR2(30),
SPRIDEN_SEARCH_LAST_NAME VARCHAR2(60),
CONVERT_SEARCH_LAST_NAME VARCHAR2(60),
SPRIDEN_SEARCH_FIRST_NAME VARCHAR2(15),
CONVERT_SEARCH_FIRST_NAME VARCHAR2(15),
SPRIDEN_SEARCH_MI VARCHAR2(15),
CONVERT_SEARCH_MI VARCHAR2(15),
SPRIDEN_SOUNDEX_LAST_NAME CHAR(4),
CONVERT_SOUNDEX_LAST_NAME VARCHAR2(4),
SPRIDEN_SOUNDEX_FIRST_NAME CHAR(4),
CONVERT_SOUNDEX_FIRST_NAME VARCHAR2(4),
SPRIDEN_NTYP_CODE VARCHAR2(4),
CONVERT_NTYP_CODE VARCHAR2(4),
SPRIDEN_CVT_RECORD_ID NUMBER(8) CONSTRAINT PK_SPRIDEN_CVT
PRIMARY KEY,
SPRIDEN_CVT_STATUS VARCHAR2(1),
SPRIDEN_CVT_JOB_ID NUMBER(8) )
STORAGE (INITIAL 1M
      PCTINCREASE 100
      MAXEXTENTS 50);
```

Caution! The script includes a DROP TABLE statement. The table (including data) is removed. Run this script only if the data in the convert table is not needed.

Note: Each convert table contains three other columns in addition to the convert columns.

1. SPRIDEN_CVT_RECORD_ID – holds a sequence number which identifies each row in the convert table.
2. SPRIDEN_CVT_STATUS -- holds the record status (New, Converted, or Error) assigned during the conversion process. It is initially populated with the value of "N" to indicate that the record is (N)ew and has not yet been processed. This value will be updated when the convert script is run in the later stages of the process. Other valid values for this field during the convert process are (C)onverted, and (E)rrored. A record with a status of (C)onverted has been successfully moved to the <*TargetTable*>. A record with a status of (E) has one or more errors that must be corrected before movement to the <*TargetTable*> can occur. An (E)rrored record will become a (C)onverted record after the error(s) have been corrected and the record has been processed via the convert script.
3. SPRIDEN_CVT_JOB_ID column is populated with a unique sequence number for each run of the conversion process. This job number is used in naming the log files that are created for each run of the convert script, allowing the log files not to be overwritten each time the convert process is run.

spraddr_cvt_create.sql

```
/*
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   09/30/2002
  This script creates the temporary conversion table SPRADDR for
  converting legacy data into SPRADDR.  This script was generated
  by the SCT Converter Tool.

AUDIT TRAIL END
*/
DROP TABLE SPRADDR_CVT;
CREATE TABLE SPRADDR_CVT
(
SPRADDR_PIDM NUMBER(8),
CONVERT_PIDM VARCHAR2(9),
SPRADDR_ATYP_CODE VARCHAR2(2),
CONVERT_ATYP_CODE VARCHAR2(2),
SPRADDR_SEQNO NUMBER(2),
CONVERT_SEQNO VARCHAR2(2),
SPRADDR_FROM_DATE DATE,
CONVERT_FROM_DATE VARCHAR2(9),
SPRADDR_TO_DATE DATE,
CONVERT_TO_DATE VARCHAR2(9),
SPRADDR_STREET_LINE1 VARCHAR2(30),
CONVERT_STREET_LINE1 VARCHAR2(30),
SPRADDR_STREET_LINE2 VARCHAR2(30),
CONVERT_STREET_LINE2 VARCHAR2(30),
SPRADDR_STREET_LINE3 VARCHAR2(30),
CONVERT_STREET_LINE3 VARCHAR2(30),
SPRADDR_CITY VARCHAR2(20),
CONVERT_CITY VARCHAR2(20),
SPRADDR_STAT_CODE VARCHAR2(3),
```

```
CONVERT_STAT_CODE VARCHAR2(3),
SPRADDR_ZIP VARCHAR2(10),
CONVERT_ZIP VARCHAR2(10),
SPRADDR_CNTY_CODE VARCHAR2(5),
CONVERT_CNTY_CODE VARCHAR2(5),
SPRADDR_NATN_CODE VARCHAR2(5),
CONVERT_NATN_CODE VARCHAR2(5),
SPRADDR_PHONE_AREA VARCHAR2(3),
CONVERT_PHONE_AREA VARCHAR2(3),
SPRADDR_PHONE_NUMBER VARCHAR2(7),
CONVERT_PHONE_NUMBER VARCHAR2(7),
SPRADDR_PHONE_EXT VARCHAR2(4),
CONVERT_PHONE_EXT VARCHAR2(4),
SPRADDR_STATUS_IND VARCHAR2(1),
CONVERT_STATUS_IND VARCHAR2(1),
SPRADDR_ACTIVITY_DATE DATE,
CONVERT_ACTIVITY_DATE VARCHAR2(9),
SPRADDR_USER VARCHAR2(30),
CONVERT_USER VARCHAR2(30),
SPRADDR_ASRC_CODE VARCHAR2(4),
CONVERT_ASRC_CODE VARCHAR2(4),
SPRADDR_DELIVERY_POINT NUMBER(2),
CONVERT_DELIVERY_POINT VARCHAR2(2),
SPRADDR_CORRECTION_DIGIT NUMBER(1),
CONVERT_CORRECTION_DIGIT VARCHAR2(1),
SPRADDR_CARRIER_ROUTE VARCHAR2(4),
CONVERT_CARRIER_ROUTE VARCHAR2(4),
SPRADDR_GST_TAX_ID VARCHAR2(15),
CONVERT_GST_TAX_ID VARCHAR2(15),
SPRADDR_REVIEWED_IND VARCHAR2(1),
CONVERT_REVIEWED_IND VARCHAR2(1),
SPRADDR_REVIEWED_USER VARCHAR2(30),
CONVERT_REVIEWED_USER VARCHAR2(30),
SPRADDR_CVT_RECORD_ID NUMBER(8) CONSTRAINT PK_SPRADDR_CVT
PRIMARY KEY,
SPRADDR_CVT_STATUS VARCHAR2(1),
SPRADDR_CVT_JOB_ID NUMBER(8) )
STORAGE (INITIAL 1M
      PCTINCREASE 100
      MAXEXTENTS 50);
```

spbpers_cvt_create.sql

```
/*
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT     09/30/2002
  This script creates the temporary conversion table SPBPERS for
  converting legacy data into SPBPERS.  This script was generated
  by the SCT Converter Tool.

AUDIT TRAIL END
*/
DROP TABLE SPBPERS_CVT;
CREATE TABLE SPBPERS_CVT
(
SPBPERS_PIDM NUMBER(8),
CONVERT_PIDM VARCHAR2(9),
SPBPERS_SSN VARCHAR2(9),
```

```
CONVERT_SSN VARCHAR2(9),
SPBPERS_BIRTH_DATE DATE,
CONVERT_BIRTH_DATE VARCHAR2(9),
SPBPERS_LGCY_CODE VARCHAR2(1),
CONVERT_LGCY_CODE VARCHAR2(1),
SPBPERS_ETHN_CODE VARCHAR2(2),
CONVERT_ETHN_CODE VARCHAR2(2),
SPBPERS_MRTL_CODE VARCHAR2(1),
CONVERT_MRTL_CODE VARCHAR2(1),
SPBPERS_RELG_CODE VARCHAR2(2),
CONVERT_RELG_CODE VARCHAR2(2),
SPBPERS_SEX VARCHAR2(1),
CONVERT_SEX VARCHAR2(1),
SPBPERS_CONFID_IND VARCHAR2(1),
CONVERT_CONFID_IND VARCHAR2(1),
SPBPERS_DEAD_IND VARCHAR2(1),
CONVERT_DEAD_IND VARCHAR2(1),
SPBPERS_VETC_FILE_NUMBER VARCHAR2(10),
CONVERT_VETC_FILE_NUMBER VARCHAR2(10),
SPBPERS_LEGAL_NAME VARCHAR2(60),
CONVERT_LEGAL_NAME VARCHAR2(60),
SPBPERS_PREF_FIRST_NAME VARCHAR2(15),
CONVERT_PREF_FIRST_NAME VARCHAR2(15),
SPBPERS_NAME_PREFIX VARCHAR2(20),
CONVERT_NAME_PREFIX VARCHAR2(20),
SPBPERS_NAME_SUFFIX VARCHAR2(20),
CONVERT_NAME_SUFFIX VARCHAR2(20),
SPBPERS_ACTIVITY_DATE DATE,
CONVERT_ACTIVITY_DATE VARCHAR2(9),
SPBPERS_VERA_IND VARCHAR2(1),
CONVERT_VERA_IND VARCHAR2(1),
SPBPERS_CITZ_IND VARCHAR2(1),
CONVERT_CITZ_IND VARCHAR2(1),
SPBPERS_DEAD_DATE DATE,
CONVERT_DEAD_DATE VARCHAR2(9),
SPBPERS_PIN RAW(1),
CONVERT_PIN VARCHAR2(1),
SPBPERS_CITZ_CODE VARCHAR2(2),
CONVERT_CITZ_CODE VARCHAR2(2),
SPBPERS_HAIR_CODE VARCHAR2(2),
CONVERT_HAIR_CODE VARCHAR2(2),
SPBPERS_EYES_CODE VARCHAR2(2),
CONVERT_EYES_CODE VARCHAR2(2),
SPBPERS_CITY_BIRTH VARCHAR2(20),
CONVERT_CITY_BIRTH VARCHAR2(20),
SPBPERS_STAT_CODE_BIRTH VARCHAR2(3),
CONVERT_STAT_CODE_BIRTH VARCHAR2(3),
SPBPERS_DRIVER_LICENSE VARCHAR2(20),
CONVERT_DRIVER_LICENSE VARCHAR2(20),
SPBPERS_STAT_CODE_DRIVER VARCHAR2(3),
CONVERT_STAT_CODE_DRIVER VARCHAR2(3),
SPBPERS_NATN_CODE_DRIVER VARCHAR2(5),
CONVERT_NATN_CODE_DRIVER VARCHAR2(5),
SPBPERS_UOMS_CODE_HEIGHT VARCHAR2(4),
CONVERT_UOMS_CODE_HEIGHT VARCHAR2(4),
SPBPERS_HEIGHT NUMBER(2),
CONVERT_HEIGHT VARCHAR2(2),
```

```
SPBPERS_UOMS_CODE_WEIGHT VARCHAR2(4),
CONVERT_UOMS_CODE_WEIGHT VARCHAR2(4),
SPBPERS_WEIGHT NUMBER(4),
CONVERT_WEIGHT VARCHAR2(4),
SPBPERS_SDVET_IND VARCHAR2(1),
CONVERT_SDVET_IND VARCHAR2(1),
SPBPERS_LICENSE_ISSUED_DATE DATE,
CONVERT_LICENSE_ISSUED_DATE VARCHAR2(9),
SPBPERS_LICENSE_EXPIRES_DATE DATE,
CONVERT_LICENSE_EXPIRES_DATE VARCHAR2(9),
SPBPERS_INCAR_IND VARCHAR2(1),
CONVERT_INCAR_IND VARCHAR2(1),
SPBPERS_WEBID RAW(1),
CONVERT_WEBID VARCHAR2(1),
SPBPERS_WEB_LAST_ACCESS RAW(1),
CONVERT_WEB_LAST_ACCESS VARCHAR2(1),
SPBPERS_PIN_DISABLED_IND RAW(1),
CONVERT_PIN_DISABLED_IND VARCHAR2(1),
SPBPERS_ITIN NUMBER(9),
CONVERT_ITIN VARCHAR2(9),
SPBPERS_CVT_RECORD_ID NUMBER(8) CONSTRAINT PK_SPBPERS_CVT
PRIMARY KEY,
SPBPERS_CVT_STATUS VARCHAR2(1),
SPBPERS_CVT_JOB_ID NUMBER(8) )
STORAGE (INITIAL 1M
      PCTINCREASE 100
      MAXEXTENTS 50);
```

Appendix C - Sample SQL*Loader Scripts

Sample <TargetTable>_cvt.ctl scripts C-3
spriden_cvt.ctl C-3
spraddr_cvt.ctl C-4
spbpers_cvt.ctl C-4

Sample <*TargetTable*>_cvt.ctl scripts

SPRIDEN: Identification Table

spriden_cvt.ctl

```
-- AUDIT TRAIL
-- Converter Tool: 2.10                SCTCVT   09/30/2002
--   This SQLLoader control file is used to load legacy data into
--   the temporary conversion table SPRIDEN_cvt.  This script
--   was generated by the SCT Converter Tool.
--
-- AUDIT TRAIL END
Load data
Infile 'spriden_cvt.dat'
APPEND
INTO TABLE SPRIDEN_CVT
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' TRAILING
NULLCOLS
(
CONVERT_PIDM CHAR,
CONVERT_ID CHAR,
CONVERT_LAST_NAME CHAR,
CONVERT_FIRST_NAME CHAR,
CONVERT_MI CHAR,
CONVERT_CHANGE_IND CHAR,
SPRIDEN_CVT_RECORD_ID SEQUENCE(MAX,1),
SPRIDEN_CVT_STATUS CONSTANT 'N'
)
```

Notice that the control file contains the *convert* columns and not the SPRIDEN columns.

Also note the name of the "Infile," *spriden_cvt.dat*. Be sure to following this naming convention in naming your flat files.

The *convert_activity_date* field can be populated with the system date or with legacy data. If legacy data exists, the user should check the "Load" and "Insert" boxes on CUACNVT for the activity date column. If legacy data does not exist, the user should leave the "Load" box blank and check only the "Insert" box so that *sysdate* will be defaulted.

SPRADDR—Address Table

spraddr_cvt.ctl

```
-- AUDIT TRAIL
-- Converter Tool:  2.10                SCTCVT   09/30/2002
--    This SQLLoader control file is used to load legacy data into
--    the temporary conversion table SPRADDR_cvt.  This script
--    was generated by the SCT Converter Tool.
--
-- AUDIT TRAIL END
Load data
Infile 'spraddr_cvt.dat'
APPEND
INTO TABLE SPRADDR_CVT
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' TRAILING NULLCOLS
(
CONVERT_PIDM CHAR,
CONVERT_ATYP_CODE CHAR,
CONVERT_FROM_DATE CHAR,
CONVERT_TO_DATE CHAR,
CONVERT_STREET_LINE1 CHAR,
CONVERT_CITY CHAR,
CONVERT_STAT_CODE CHAR,
CONVERT_ZIP CHAR,
SPRADDR_CVT_RECORD_ID SEQUENCE(MAX,1),
SPRADDR_CVT_STATUS CONSTANT 'N'
)
```

Note that the *convert_pidm* column remains in the control file. This field will be loaded with the first value in the client flat file, which is the legacy ID (SSN in this example). Recall in the sample conversion that the column length was modified from 8 to 9 on CUACNVT. This allows the SQL*Loader utility to load the legacy ID, with a length of 9, into the *convert_pidm* column. During the conversion process, through the use of a function, this ID will be used to match against the *spriden_id*, which at this point in the conversion process, must already have been loaded.

SPBPERS General Person Biographic Demographic Table

spbpers_cvt.ctl

```
-- AUDIT TRAIL
-- Converter Tool:  2.10                SCTCVT   09/30/2002
--    This SQLLoader control file is used to load legacy data into
--    the temporary conversion table SPBPERS_cvt.  This script
--    was generated by the SCT Converter Tool.
--
-- AUDIT TRAIL END
Load data
Infile 'spbpers_cvt.dat'
APPEND
INTO TABLE SPBPERS_CVT
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' TRAILING
NULLCOLS
(
CONVERT_PIDM CHAR,
CONVERT_SSN CHAR,
CONVERT_BIRTH_DATE CHAR,
```

```
CONVERT_MRTL_CODE CHAR,
CONVERT_ETHN_CODE CHAR,
CONVERT_SEX CHAR,
SPBPERS_CVT_RECORD_ID SEQUENCE(MAX,1),
SPBPERS_CVT_STATUS CONSTANT 'N'
)
```

Appendix D - Sample Conversion Scripts

Sample <TargetTable>_convert.sql scripts D-3
spriden_convert.sql D-3
SPRADDR—Address Table D-17
spraddr_convert.sql D-17
SPBPERS: General Person Biographic Demographic Table D-34
spbpers_convert.sql D-34

Sample <TargetTable>_convert.sql scripts

This is the entire text of the 3 *<TargetTable>_convert.sql* scripts used in our sample conversion. They were created from the SCT Converter Tool after specifications were entered on the CUACNVT form. Areas of interest are highlighted and/or have notes in the right margin.

SPRIDEN: Identification Table

spriden_convert.sql

```
/*
================================================
Process: spriden_convert.sql
Generated: 30-Sep-2002 09:27
================================================
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   09/30/2002
  This script translates legacy values into Banner values and
  inserts Banner values into spriden.  This script was
  generated by the SCT Converter Tool.

AUDIT TRAIL END
*/
COLUMN file_id NEW_VALUE spool_file NOPRINT;
SET VERIFY OFF;
SET ECHO OFF;
SELECT 'spriden_'||cubcnvt_sequence.NEXTVAL||'.log' file_id FROM DUAL;
SPOOL &spool_file;
PROMPT
PROMPT Please enter the record type which should be processed.
PROMPT  (N)ew records
PROMPT  (C)onverted records
PROMPT  (E)rrored records
ACCEPT records_in CHAR PROMPT 'Include:'
PROMPT
PROMPT Please enter the disposition to which the records should
PROMPT be processed.
PROMPT  (C)onvert records only
PROMPT  (I)nsert records only (records were previously converted successfully).
PROMPT  (B)oth convert and insert records into spriden
ACCEPT process_level CHAR PROMPT 'Disposition: '
PROMPT
SET SERVEROUTPUT ON SIZE 500000;
PROMPT
PROMPT ================================================
PROMPT = will be creating generated id records       =
PROMPT = in SPRIDEN_CVT for any converted records    =
PROMPT =  (i.e. records with spriden_cvt_status = 'C' =
PROMPT ================================================
PROMPT
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') START_TIME FROM DUAL;
DECLARE
  cur_jobid NUMBER := 0;
  cur_owner VARCHAR2(30) := 'SATURN';
  cur_tbl VARCHAR2(30) := 'SPRIDEN';
  cur_fileid VARCHAR2(2000);
  col_status  VARCHAR2(1);
```

```
  legacy_value VARCHAR2(2000);
  process_status VARCHAR2(1) := 'C';
  err_num  NUMBER(5);
  col_err VARCHAR2(100);
  cur_rec  NUMBER(8);
  cur_col VARCHAR2(30);
  rows_errored NUMBER(8) := 0;
  rows_success_convert NUMBER(8) := 0;
  rows_success_insert NUMBER(8) := 0;
  commit_counter NUMBER(8);
  commit_frequency NUMBER(8):= 250;
  converted_val VARCHAR2(2000);
  err_msg  VARCHAR2(300);
  first_row BOOLEAN := TRUE;
  column_error EXCEPTION;
  process_level VARCHAR2(1) := UPPER('&process_level');
  records_in VARCHAR2(1) := UPPER('&records_in');
  break1_value VARCHAR2(30);
  break2_value VARCHAR2(30);
  break3_value VARCHAR2(30);
  seq1 NUMBER(5) := 0;
  seq2 NUMBER(5) := 0;
  seq3 NUMBER(5) := 0;
  CURSOR spriden_cursor IS
    SELECT * FROM spriden_CVT
     WHERE spriden_cvt_status = records_in

     order by spriden_cvt_record_id;
--     order by convert_change_ind desc

--NOTE: THIS IS THE ORDER BY STATEMENT THAT MUST BE ALTERED IN
ORDER FOR THE F_CVT_AUTO_PIDM FUNCTION TO PERFORM PROPERLY.

BEGIN
  IF process_level = 'I' AND
     records_in <> 'C' THEN
     DBMS_OUTPUT.PUT_LINE('If inserting only, you must choose to include converted
records only.');
     GOTO end_of_program;
  END IF;
  SELECT cubcnvt_sequence.CURRVAL INTO cur_jobid FROM DUAL;
  DBMS_OUTPUT.ENABLE(100000);
  DBMS_OUTPUT.PUT_LINE('Beginning Conversion of Table spriden');
  DBMS_OUTPUT.PUT_LINE('Job Number: '||cur_jobid);
  commit_counter := 0;
  <<spriden_loop>>
  FOR spriden_rec IN spriden_cursor LOOP
  BEGIN
    cur_rec := spriden_rec.spriden_cvt_record_id;
    col_status := 'C';
    DELETE FROM curcerr
     WHERE curcerr_table_owner = cur_owner
       AND curcerr_table_name = cur_tbl
       AND curcerr_record_id = cur_rec;
```

```
IF process_level = 'I' THEN
      GOTO insert_spriden;
    END IF;
    /*-------------------------------
    |Beginning evaluation of column
    |SPRIDEN_PIDM
    ---------------------------------*/
    BEGIN
      cur_col := 'SPRIDEN_PIDM';
      legacy_value := spriden_rec.convert_pidm;
      /* CURCNVT Rules:
        Column is Required
        Conversion function is specified
        No validation specified
        No default value
      */
      converted_val := null;
      converted_val := F_CVT_AUTO_PIDM(LEGACY_VALUE);
      IF SUBSTR(converted_val,1,3) = 'ERR' OR
        SUBSTR(converted_val,1,3) = 'ORA' THEN
        err_msg := 'Column conv failure:'||converted_val;
        RAISE column_error;
      END IF;
      IF converted_val IS NULL THEN
          err_msg := 'Missing required value';
          RAISE column_error;
      END IF;
      BEGIN
        spriden_rec.SPRIDEN_PIDM := TO_NUMBER(converted_val
        );
      EXCEPTION
        WHEN OTHERS THEN
          err_msg := 'Invalid Data Format';
          RAISE column_error;
      END;
    EXCEPTION
      WHEN column_error THEN
        col_status := 'E';
        process_status := 'E';
        p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
        legacy_value,converted_val,null,err_msg);
    END;
    /*-------------------------------
    |Beginning evaluation of column
    |SPRIDEN_ID
    ---------------------------------*/
    BEGIN
      cur_col := 'SPRIDEN_ID';
      legacy_value := spriden_rec.convert_id;
      /* CURCNVT Rules:
        Column is Required
        Conversion function is not specified
        No validation specified
        No default value
      */
      converted_val := null;
      converted_val := spriden_rec.CONVERT_ID;
```

```
   IF converted_val IS NULL THEN
        err_msg := 'Missing required value';
        RAISE column_error;
   END IF;
   spriden_rec.SPRIDEN_ID := converted_val;
EXCEPTION
   WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_LAST_NAME
---------------------------------*/
BEGIN
   cur_col := 'SPRIDEN_LAST_NAME';
   legacy_value := spriden_rec.convert_last_name;
   /* CURCNVT Rules:
     Column is Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spriden_rec.CONVERT_LAST_NAME;
   IF converted_val IS NULL THEN
        err_msg := 'Missing required value';
        RAISE column_error;
   END IF;
   spriden_rec.SPRIDEN_LAST_NAME := converted_val;
EXCEPTION
   WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_FIRST_NAME
---------------------------------*/
BEGIN
   cur_col := 'SPRIDEN_FIRST_NAME';
   legacy_value := spriden_rec.convert_first_name;
   /* CURCNVT Rules:
     Column is Not Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spriden_rec.CONVERT_FIRST_NAME;
   spriden_rec.SPRIDEN_FIRST_NAME := converted_val;
EXCEPTION
```

```
 WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_MI
---------------------------------*/
BEGIN
  cur_col := 'SPRIDEN_MI';
  legacy_value := spriden_rec.convert_mi;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_MI;
  spriden_rec.SPRIDEN_MI := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_CHANGE_IND
---------------------------------*/
BEGIN
  cur_col := 'SPRIDEN_CHANGE_IND';
  legacy_value := spriden_rec.convert_change_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_CHANGE_IND;
  spriden_rec.SPRIDEN_CHANGE_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_ENTITY_IND
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPRIDEN_ENTITY_IND';
  legacy_value := spriden_rec.convert_entity_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_ENTITY_IND;
  spriden_rec.SPRIDEN_ENTITY_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRIDEN_ACTIVITY_DATE
--------------------------------*/
BEGIN
  cur_col := 'SPRIDEN_ACTIVITY_DATE';
  legacy_value := spriden_rec.convert_activity_date;
  /* CURCNVT Rules:
    Column is Required
    Conversion function is specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := SYSDATE;  -- DEFAULTING SYSDATE HERE
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  BEGIN
    spriden_rec.SPRIDEN_ACTIVITY_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
```

```
    END;
     /*-------------------------------
     |Beginning evaluation of column
     |SPRIDEN_USER
     ---------------------------------*/
     BEGIN
       cur_col := 'SPRIDEN_USER';
       legacy_value := spriden_rec.convert_user;
       /* CURCNVT Rules:
         Column is Not Required
         Conversion function is not specified
         No validation specified
         Default value 'SCTCVT'
         Default Action 'D' - Default if null legacy value  -- DEFINING DEFAULT USER
AND DEFAULT ACTION
       */
       converted_val := null;
       converted_val := NVL(spriden_rec.CONVERT_USER,'SCTCVT');
       spriden_rec.SPRIDEN_USER := converted_val;
     EXCEPTION
       WHEN column_error THEN
         col_status := 'E';
         process_status := 'E';
         p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
         legacy_value,converted_val,null,err_msg);
     END;
     /*-------------------------------
     |Beginning evaluation of column
     |SPRIDEN_ORIGIN
     ---------------------------------*/
     BEGIN
       cur_col := 'SPRIDEN_ORIGIN';
       legacy_value := spriden_rec.convert_origin;
       /* CURCNVT Rules:
         Column is Not Required
         Conversion function is not specified
         No validation specified
         Default value 'CONVERSION'
         Default Action 'D' - Default if null legacy value
       */
       converted_val := null;
       converted_val := NVL(spriden_rec.CONVERT_ORIGIN,'CONVERSION');
       spriden_rec.SPRIDEN_ORIGIN := converted_val;
     EXCEPTION
       WHEN column_error THEN
         col_status := 'E';
         process_status := 'E';
         p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
         legacy_value,converted_val,null,err_msg);
     END;
     /*-------------------------------
     |Beginning evaluation of column
     |SPRIDEN_SEARCH_LAST_NAME
     ---------------------------------*/
```

```
 BEGIN
   cur_col := 'SPRIDEN_SEARCH_LAST_NAME';
   legacy_value := spriden_rec.convert_search_last_name;
   /* CURCNVT Rules:
     Column is Not Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spriden_rec.CONVERT_SEARCH_LAST_NAME;
   spriden_rec.SPRIDEN_SEARCH_LAST_NAME := converted_val;
EXCEPTION
   WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRIDEN_SEARCH_FIRST_NAME
--------------------------------*/
BEGIN
   cur_col := 'SPRIDEN_SEARCH_FIRST_NAME';
   legacy_value := spriden_rec.convert_search_first_name;
   /* CURCNVT Rules:
     Column is Not Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spriden_rec.CONVERT_SEARCH_FIRST_NAME;
   spriden_rec.SPRIDEN_SEARCH_FIRST_NAME := converted_val;
EXCEPTION
   WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRIDEN_SEARCH_MI
--------------------------------*/
BEGIN
   cur_col := 'SPRIDEN_SEARCH_MI';
   legacy_value := spriden_rec.convert_search_mi;
   /* CURCNVT Rules:
     Column is Not Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spriden_rec.CONVERT_SEARCH_MI;
   spriden_rec.SPRIDEN_SEARCH_MI := converted_val;
```

```
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_SOUNDEX_LAST_NAME
---------------------------------*/
BEGIN
  cur_col := 'SPRIDEN_SOUNDEX_LAST_NAME';
  legacy_value := spriden_rec.convert_soundex_last_name;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_SOUNDEX_LAST_NAME;
  spriden_rec.SPRIDEN_SOUNDEX_LAST_NAME := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_SOUNDEX_FIRST_NAME
---------------------------------*/
BEGIN
  cur_col := 'SPRIDEN_SOUNDEX_FIRST_NAME';
  legacy_value := spriden_rec.convert_soundex_first_name;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_SOUNDEX_FIRST_NAME;
  spriden_rec.SPRIDEN_SOUNDEX_FIRST_NAME := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRIDEN_NTYP_CODE
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPRIDEN_NTYP_CODE';
  legacy_value := spriden_rec.convert_ntyp_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spriden_rec.CONVERT_NTYP_CODE;
  spriden_rec.SPRIDEN_NTYP_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-----------------------------------------------------
| Updating Temporary table with converted values
-----------------------------------------------------*/
BEGIN
  UPDATE SPRIDEN_CVT
    SET
    SPRIDEN_PIDM = spriden_rec.spriden_pidm,
    SPRIDEN_ID = spriden_rec.spriden_id,
    SPRIDEN_LAST_NAME = spriden_rec.spriden_last_name,
    SPRIDEN_FIRST_NAME = spriden_rec.spriden_first_name,
    SPRIDEN_MI = spriden_rec.spriden_mi,
    SPRIDEN_CHANGE_IND = spriden_rec.spriden_change_ind,
    SPRIDEN_ENTITY_IND = spriden_rec.spriden_entity_ind,
    SPRIDEN_ACTIVITY_DATE = SYSDATE,
    SPRIDEN_USER = spriden_rec.spriden_user,
    SPRIDEN_ORIGIN = spriden_rec.spriden_origin,
    SPRIDEN_SEARCH_LAST_NAME = spriden_rec.spriden_search_last_name,
    SPRIDEN_SEARCH_FIRST_NAME = spriden_rec.spriden_search_first_name,
    SPRIDEN_SEARCH_MI = spriden_rec.spriden_search_mi,
    SPRIDEN_SOUNDEX_LAST_NAME = spriden_rec.spriden_soundex_last_name,
    SPRIDEN_SOUNDEX_FIRST_NAME = spriden_rec.spriden_soundex_first_name,
    SPRIDEN_NTYP_CODE = spriden_rec.spriden_ntyp_code,
    SPRIDEN_CVT_STATUS = col_status,
    SPRIDEN_CVT_JOB_ID = cur_jobid
    WHERE SPRIDEN_CVT_RECORD_ID = cur_rec;
    rows_success_convert := rows_success_convert + 1;
EXCEPTION
  WHEN OTHERS THEN
    err_msg := 'Error updating record';
    RAISE;
END;
<<insert_spriden>>
IF process_level = 'B'
OR process_level = 'I' THEN
  /*---------------------------------------
  |Inserting into Banner Table spriden
  ---------------------------------------*/
```

```
    BEGIN
      INSERT INTO SATURN.spriden
      (
      SPRIDEN_PIDM
      ,SPRIDEN_ID
      ,SPRIDEN_LAST_NAME
      ,SPRIDEN_FIRST_NAME
      ,SPRIDEN_MI
      ,SPRIDEN_CHANGE_IND
      ,SPRIDEN_ACTIVITY_DATE
      ,SPRIDEN_USER
      ,SPRIDEN_ORIGIN
      )
      VALUES(
      spriden_rec.SPRIDEN_PIDM
      ,spriden_rec.SPRIDEN_ID
      ,spriden_rec.SPRIDEN_LAST_NAME
      ,spriden_rec.SPRIDEN_FIRST_NAME
      ,spriden_rec.SPRIDEN_MI
      ,spriden_rec.SPRIDEN_CHANGE_IND
      ,spriden_rec.SPRIDEN_ACTIVITY_DATE
      ,spriden_rec.SPRIDEN_USER
      ,spriden_rec.SPRIDEN_ORIGIN
      );
      rows_success_insert := rows_success_insert + 1;
      UPDATE SPRIDEN_CVT
        SET spriden_cvt_status = 'I'
      WHERE SPRIDEN_CVT_RECORD_ID = cur_rec;
    EXCEPTION
      WHEN OTHERS THEN
        err_msg := 'Error inserting into Banner table';
        col_status := 'E';
        cur_col := NULL;
        RAISE;
    END;
  END IF;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
  WHEN OTHERS THEN
    err_num := SQLCODE;
    err_msg := SUBSTR(SQLERRM, 1, 100);
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    null,null,null,err_msg);
  END;
IF col_status = 'E' THEN
  rows_errored := rows_errored + 1;
END IF;
if commit_counter = commit_frequency then
  commit;
  commit_counter := 0;
else
  commit_counter := commit_counter + 1;
end if;
END LOOP spriden_loop;
```

```
-- **************************************************************************
-- function to create spriden records in convert table with generated IDs
-- **************************************************************************
BEGIN
DECLARE
CURSOR max_record_nbr IS
  SELECT max(spriden_cvt_record_id)
    FROM spriden_cvt;
max_rec_nbr   spriden_cvt.spriden_cvt_record_id%TYPE;
CURSOR PTI_CURSOR IS
  SELECT SOBSEQN_SEQNO_PREFIX || LPAD(TO_CHAR(
       SOBSEQN_MAXSEQNO), 8, '0')
  FROM   SOBSEQN
 WHERE  SOBSEQN_FUNCTION = 'ID';
 new_id VARCHAR2(9);
 gen_id_cnt INTEGER;
-- cursor only gets spriden_cvt records that do not have
-- an ID generated already, i.e. those PIDMS where there
-- is not a record with a ntype code LGCY and change indicator of I
-- in SPRIDEN_CVT or the actual BANNER SPRIDEN table
CURSOR SPRIDEN_CVT_RECORDS IS
   SELECT *
    FROM spriden_cvt x
   WHERE spriden_cvt_status = 'C'
    AND spriden_change_ind IS NULL
          AND not exists
            (SELECT 'X' from spriden_cvt y
              where x.spriden_pidm = y.spriden_pidm
                    and spriden_change_ind = 'I'
                    and spriden_ntyp_code = 'LGCY');
   cvt_rec      SPRIDEN_CVT_RECORDS%rowtype;
   CURSOR SPRIDEN_L_RECORDS IS
       SELECT rowid
      FROM spriden_cvt
     WHERE spriden_pidm = cvt_rec.spriden_pidm
      AND spriden_change_ind = 'I'
      AND spriden_ntyp_code = 'LGCY';
   l_record           SPRIDEN_L_RECORDS%rowtype;
   l_record_count      INTEGER;
--
BEGIN
 gen_id_cnt := 0;
-- get max record number
 OPEN max_record_nbr;
 FETCH max_record_nbr INTO max_rec_nbr;
 CLOSE max_record_nbr;
--  select all the converted records where the change_ind is null
--  and ID not already generated
 OPEN SPRIDEN_CVT_RECORDS;
 LOOP
   FETCH SPRIDEN_CVT_RECORDS INTO cvt_rec;
   IF SPRIDEN_CVT_RECORDS%NOTFOUND THEN
     EXIT;
   END IF;
```

```
--
  gen_id_cnt := gen_id_cnt + 1;
--
--    get new id number
    UPDATE SOBSEQN
     SET    SOBSEQN_MAXSEQNO = SOBSEQN_MAXSEQNO + 1,
           SOBSEQN_ACTIVITY_DATE = SYSDATE
      WHERE  SOBSEQN_FUNCTION = 'ID';
--
    OPEN PTI_CURSOR ;
    FETCH PTI_CURSOR INTO new_id;
    CLOSE PTI_CURSOR;
--
--  update current record with change indicator of I and ntyp code LGCY
--
    UPDATE spriden_cvt
      SET spriden_change_ind = 'I'
         ,spriden_ntyp_code = 'LGCY'
     WHERE spriden_pidm = cvt_rec.spriden_pidm
       and spriden_change_ind IS NULL;
---
    max_rec_nbr := max_rec_nbr + 1;
    cvt_rec.spriden_id := new_id;
    cvt_rec.spriden_cvt_record_id := max_rec_nbr;
    cvt_rec.spriden_change_ind := NULL;
--
--  create record in spriden_cvt with generated ID
    INSERT INTO SPRIDEN_CVT (
 SPRIDEN_PIDM, CONVERT_PIDM
,  SPRIDEN_ID, CONVERT_ID
,  SPRIDEN_LAST_NAME, CONVERT_LAST_NAME
,  SPRIDEN_FIRST_NAME, CONVERT_FIRST_NAME
,  SPRIDEN_MI, CONVERT_MI
,  SPRIDEN_CHANGE_IND, CONVERT_CHANGE_IND
,  SPRIDEN_ENTITY_IND, CONVERT_ENTITY_IND
,  SPRIDEN_ACTIVITY_DATE, CONVERT_ACTIVITY_DATE
,  SPRIDEN_USER, CONVERT_USER
,  SPRIDEN_ORIGIN, CONVERT_ORIGIN
,  SPRIDEN_SEARCH_LAST_NAME, CONVERT_SEARCH_LAST_NAME
,  SPRIDEN_SEARCH_FIRST_NAME, CONVERT_SEARCH_FIRST_NAME
,  SPRIDEN_SEARCH_MI, CONVERT_SEARCH_MI
,  SPRIDEN_SOUNDEX_LAST_NAME, CONVERT_SOUNDEX_LAST_NAME
,  SPRIDEN_SOUNDEX_FIRST_NAME, CONVERT_SOUNDEX_FIRST_NAME
,  SPRIDEN_NTYP_CODE, CONVERT_NTYP_CODE
,  SPRIDEN_CVT_STATUS, SPRIDEN_CVT_JOB_ID
, SPRIDEN_CVT_RECORD_ID)
  VALUES (
 cvt_rec.SPRIDEN_PIDM, cvt_rec.CONVERT_PIDM
,   new_id, cvt_rec.CONVERT_ID
,  cvt_rec.SPRIDEN_LAST_NAME, cvt_rec.CONVERT_LAST_NAME
,  cvt_rec.SPRIDEN_FIRST_NAME, cvt_rec.CONVERT_FIRST_NAME
,  cvt_rec.SPRIDEN_MI, cvt_rec.CONVERT_MI
,  NULL, cvt_rec.CONVERT_CHANGE_IND
,  cvt_rec.SPRIDEN_ENTITY_IND, cvt_rec.CONVERT_ENTITY_IND
,  cvt_rec.SPRIDEN_ACTIVITY_DATE, cvt_rec.CONVERT_ACTIVITY_DATE
,  cvt_rec.SPRIDEN_USER, cvt_rec.CONVERT_USER
```

```
  ,  cvt_rec.SPRIDEN_ORIGIN, cvt_rec.CONVERT_ORIGIN
  ,  cvt_rec.SPRIDEN_SEARCH_LAST_NAME,
cvt_rec.CONVERT_SEARCH_LAST_NAME
  ,  cvt_rec.SPRIDEN_SEARCH_FIRST_NAME,
cvt_rec.CONVERT_SEARCH_FIRST_NAME
  ,  cvt_rec.SPRIDEN_SEARCH_MI, cvt_rec.CONVERT_SEARCH_MI
  ,  cvt_rec.SPRIDEN_SOUNDEX_LAST_NAME,
cvt_rec.CONVERT_SOUNDEX_LAST_NAME
  , cvt_rec.SPRIDEN_SOUNDEX_FIRST_NAME,
cvt_rec.CONVERT_SOUNDEX_FIRST_NAME
  ,cvt_rec.SPRIDEN_NTYP_CODE, cvt_rec.CONVERT_NTYP_CODE
  ,cvt_rec.SPRIDEN_CVT_STATUS, cvt_rec.SPRIDEN_CVT_JOB_ID
     , cvt_rec.SPRIDEN_CVT_RECORD_ID) ;
  -- end of loop getting spriden_cvt records that do not have generated id
  if commit_counter = commit_frequency then
    commit;
    commit_counter := 0;
  else
    commit_counter := commit_counter + 3;
  end if;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Number of ids generated = '|| to_char(gen_id_cnt));
  -- end of generated id code
  -- end of generated id code
  END;
  -- end of generated id block
  END;
DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPRIDEN_CVT:
'||rows_success_convert);
DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spriden:
'||rows_success_insert);
DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
<<end_of_program>>
DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spriden');
EXCEPTION
WHEN OTHERS THEN
  err_num := SQLCODE;
  err_msg := SUBSTR(SQLERRM, 1, 100);
  process_status := 'E';
  p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
  null,null,null,err_msg);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPRIDEN_CVT:
'||rows_success_convert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spriden:
'||rows_success_insert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
  DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spriden');
END;
/
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') STOP_TIME FROM DUAL;
COMMIT;
PROMPT Querying CURCERR table for errors
col msg for a35 hea 'ERROR MESSAGE' WORD
col colname for a30 hea 'COLUMN_NAME'
col jobno for 99999 hea 'JOB'
col norecs for 999999 hea 'COUNT'
set wrap on
```

```
select distinct curcerr_message msg,
       curcerr_column_name colname,
       curcerr_cvt_identifier jobno,
       count(*) norecs
from curcerr
where curcerr_table_name = 'SPRIDEN'
group by curcerr_message,
curcerr_column_name,
curcerr_cvt_identifier;
SPOOL OFF
UNDEFINE process_level
UNDEFINE records_in
```

SPRADDR—Address Table

spraddr_convert.sql

```
/*
================================================
Process: spraddr_convert.sql
Generated: 30-Sep-2002 09:13
================================================
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   09/30/2002
  This script translates legacy values into Banner values and
  inserts Banner values into spraddr.  This script was
  generated by the SCT Converter Tool.

AUDIT TRAIL END
*/
COLUMN file_id NEW_VALUE spool_file NOPRINT;
SET VERIFY OFF;
SET ECHO OFF;
SELECT 'spraddr_'||cubcnvt_sequence.NEXTVAL||'.log' file_id FROM DUAL;
SPOOL &spool_file;
PROMPT
PROMPT Please enter the record type which should be processed.
PROMPT  (N)ew records
PROMPT  (C)onverted records
PROMPT  (E)rrored records
ACCEPT records_in CHAR PROMPT 'Include:'
PROMPT
PROMPT Please enter the disposition to which the records should
PROMPT be processed.
PROMPT  (C)onvert records only
PROMPT  (I)nsert records only (records were previously converted successfully).
PROMPT  (B)oth convert and insert records into spraddr
ACCEPT process_level CHAR PROMPT 'Disposition: '
PROMPT
SET SERVEROUTPUT ON SIZE 500000;
PROMPT
PROMPT ================================================
PROMPT = will be creating generated id records       =
PROMPT = in SPRIDEN_CVT for any converted records    =
PROMPT =  (i.e. records with spriden_cvt_status = 'C' =
PROMPT ================================================
PROMPT
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') START_TIME FROM DUAL;
```

```
DECLARE
  cur_jobid NUMBER := 0;
  cur_owner VARCHAR2(30) := 'SATURN';
  cur_tbl VARCHAR2(30) := 'SPRADDR';
  cur_fileid VARCHAR2(2000);
  col_status  VARCHAR2(1);
  legacy_value VARCHAR2(2000);
  process_status VARCHAR2(1) := 'C';
  err_num  NUMBER(5);
  col_err VARCHAR2(100);
  cur_rec  NUMBER(8);
  cur_col VARCHAR2(30);
  rows_errored NUMBER(8) := 0;
  rows_success_convert NUMBER(8) := 0;
  rows_success_insert NUMBER(8) := 0;
  commit_counter NUMBER(8);
  commit_frequency NUMBER(8):= 250;
  converted_val VARCHAR2(2000);
  err_msg  VARCHAR2(300);
  first_row BOOLEAN := TRUE;
  column_error EXCEPTION;
  process_level VARCHAR2(1) := UPPER('&process_level');
  records_in VARCHAR2(1) := UPPER('&records_in');
  break1_value VARCHAR2(30);
  break2_value VARCHAR2(30);
  break3_value VARCHAR2(30);
  seq1 NUMBER(5) := 0;
  seq2 NUMBER(5) := 0;
  seq3 NUMBER(5) := 0;
  CURSOR spraddr_cursor IS
    SELECT * FROM spraddr_CVT
     WHERE spraddr_cvt_status = records_in

  ORDER BY CONVERT_PIDM
  ,
        CONVERT_ATYP_CODE
  ,
        spraddr_cvt_record_id;
BEGIN
  IF process_level = 'I' AND
     records_in <> 'C' THEN
     DBMS_OUTPUT.PUT_LINE('If inserting only, you must choose to include converted records
only.');
     GOTO end_of_program;
  END IF;
  SELECT cubcnvt_sequence.CURRVAL INTO cur_jobid FROM DUAL;
  DBMS_OUTPUT.ENABLE(100000);
  DBMS_OUTPUT.PUT_LINE('Beginning Conversion of Table spraddr');
  DBMS_OUTPUT.PUT_LINE('Job Number: '||cur_jobid);
  commit_counter := 0;
  <<spraddr_loop>>
  FOR spraddr_rec IN spraddr_cursor LOOP
  BEGIN
     cur_rec := spraddr_rec.spraddr_cvt_record_id;
     col_status := 'C';
     DELETE FROM curcerr
      WHERE curcerr_table_owner = cur_owner
```

```
   AND curcerr_table_name = cur_tbl
   AND curcerr_record_id = cur_rec;

IF process_level = 'I' THEN
  GOTO insert_spraddr;
END IF;
/*-------------------------------------------------
| Checking for Break Changes
-------------------------------------------------*/
IF NOT first_row THEN
  IF break2_value <> spraddr_rec.CONVERT_ATYP_CODE THEN
    err_msg := F_GEN_RESET_SEQNO(SEQ1);
    IF SUBSTR(err_msg,1,3) = 'ERR' OR
      SUBSTR(err_msg,1,3) = 'ORA' THEN
      DBMS_OUTPUT.PUT_LINE('Break 2 Error');
    END IF;
  END IF;
  IF break1_value <> spraddr_rec.CONVERT_PIDM THEN
    COMMIT;
    err_msg := F_GEN_RESET_SEQNO(SEQ1);
    IF SUBSTR(err_msg,1,3) = 'ERR' OR
      SUBSTR(err_msg,1,3) = 'ORA' THEN
      DBMS_OUTPUT.PUT_LINE('Break 1 Error');
    END IF;
  END IF;
END IF;
FIRST_ROW := FALSE;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_PIDM
-------------------------------*/
BEGIN
  cur_col := 'SPRADDR_PIDM';
  legacy_value := spraddr_rec.convert_pidm;
  /* CURCNVT Rules:
   Column is Required
   Conversion function is specified
   No validation specified
   No default value
  */
  converted_val := null;
  converted_val := F_CVT_GET_PIDM(LEGACY_VALUE);
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  BEGIN
    spraddr_rec.SPRADDR_PIDM := TO_NUMBER(converted_val
    );
```

```
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_ATYP_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_ATYP_CODE';
  legacy_value := spraddr_rec.convert_atyp_code;
  /* CURCNVT Rules:
   Column is Required
   Conversion function is specified
   No validation specified
   No default value
  */
  converted_val := null;
  converted_val := F_CVT_CURCVAL_RNULL('STVATYP',LEGACY_VALUE);
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  spraddr_rec.SPRADDR_ATYP_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_SEQNO
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_SEQNO';
  legacy_value := spraddr_rec.convert_seqno;
  /* CURCNVT Rules:
   Column is Required
   Conversion function is specified
   No validation specified
   No default value
  */
```

```
  converted_val := null;
  converted_val := F_GEN_SEQUENCE(SEQ1);
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  BEGIN
    spraddr_rec.SPRADDR_SEQNO := TO_NUMBER(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_FROM_DATE
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_FROM_DATE';
  legacy_value := spraddr_rec.convert_from_date;
  /* CURCNVT Rules:
   Column is Not Required
   Conversion function is not specified
   No validation specified
   No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_FROM_DATE;
  BEGIN
    spraddr_rec.SPRADDR_FROM_DATE := TO_DATE(converted_val
    ,'YYYYMMDD'
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
```

```
|Beginning evaluation of column
|SPRADDR_TO_DATE
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_TO_DATE';
  legacy_value := spraddr_rec.convert_to_date;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_TO_DATE;
  BEGIN
    spraddr_rec.SPRADDR_TO_DATE := TO_DATE(converted_val
    ,'YYYYMMDD'
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRADDR_STREET_LINE1
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_STREET_LINE1';
  legacy_value := spraddr_rec.convert_street_line1;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_STREET_LINE1;
  spraddr_rec.SPRADDR_STREET_LINE1 := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRADDR_STREET_LINE2
----------------------------------*/
```

```
BEGIN
  cur_col := 'SPRADDR_STREET_LINE2';
  legacy_value := spraddr_rec.convert_street_line2;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_STREET_LINE2;
  spraddr_rec.SPRADDR_STREET_LINE2 := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPRADDR_STREET_LINE3
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_STREET_LINE3';
  legacy_value := spraddr_rec.convert_street_line3;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_STREET_LINE3;
  spraddr_rec.SPRADDR_STREET_LINE3 := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPRADDR_CITY
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_CITY';
  legacy_value := spraddr_rec.convert_city;
  /* CURCNVT Rules:
    Column is Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_CITY;
```

```
  IF converted_val IS NULL THEN
       err_msg := 'Missing required value';
       RAISE column_error;
  END IF;
  spraddr_rec.SPRADDR_CITY := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_STAT_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_STAT_CODE';
  legacy_value := spraddr_rec.convert_stat_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_STAT_CODE;
  spraddr_rec.SPRADDR_STAT_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_ZIP
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_ZIP';
  legacy_value := spraddr_rec.convert_zip;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_ZIP;
  spraddr_rec.SPRADDR_ZIP := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
```

```
|Beginning evaluation of column
|SPRADDR_CNTY_CODE
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_CNTY_CODE';
  legacy_value := spraddr_rec.convert_cnty_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_CNTY_CODE;
  spraddr_rec.SPRADDR_CNTY_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_NATN_CODE
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_NATN_CODE';
  legacy_value := spraddr_rec.convert_natn_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_NATN_CODE;
  spraddr_rec.SPRADDR_NATN_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_PHONE_AREA
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_PHONE_AREA';
  legacy_value := spraddr_rec.convert_phone_area;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
```

```
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_PHONE_AREA;
  spraddr_rec.SPRADDR_PHONE_AREA := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_PHONE_NUMBER
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_PHONE_NUMBER';
  legacy_value := spraddr_rec.convert_phone_number;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_PHONE_NUMBER;
  spraddr_rec.SPRADDR_PHONE_NUMBER := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_PHONE_EXT
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_PHONE_EXT';
  legacy_value := spraddr_rec.convert_phone_ext;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_PHONE_EXT;
  spraddr_rec.SPRADDR_PHONE_EXT := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
```

```
|Beginning evaluation of column
|SPRADDR_STATUS_IND
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_STATUS_IND';
  legacy_value := spraddr_rec.convert_status_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_STATUS_IND;
  spraddr_rec.SPRADDR_STATUS_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_ACTIVITY_DATE
----------------------------------*/
BEGIN
  cur_col := 'SPRADDR_ACTIVITY_DATE';
  legacy_value := spraddr_rec.convert_activity_date;
  /* CURCNVT Rules:
    Column is Required
    Conversion function is specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := SYSDATE;
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  BEGIN
    spraddr_rec.SPRADDR_ACTIVITY_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
```

```
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRADDR_USER
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_USER';
  legacy_value := spraddr_rec.convert_user;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_USER;
  spraddr_rec.SPRADDR_USER := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRADDR_ASRC_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_ASRC_CODE';
  legacy_value := spraddr_rec.convert_asrc_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_ASRC_CODE;
  spraddr_rec.SPRADDR_ASRC_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPRADDR_DELIVERY_POINT
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_DELIVERY_POINT';
  legacy_value := spraddr_rec.convert_delivery_point;
  /* CURCNVT Rules:
```

```
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_DELIVERY_POINT;
  BEGIN
    spraddr_rec.SPRADDR_DELIVERY_POINT := TO_NUMBER(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_CORRECTION_DIGIT
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_CORRECTION_DIGIT';
  legacy_value := spraddr_rec.convert_correction_digit;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_CORRECTION_DIGIT;
  BEGIN
    spraddr_rec.SPRADDR_CORRECTION_DIGIT := TO_NUMBER(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_CARRIER_ROUTE
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPRADDR_CARRIER_ROUTE';
  legacy_value := spraddr_rec.convert_carrier_route;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_CARRIER_ROUTE;
  spraddr_rec.SPRADDR_CARRIER_ROUTE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPRADDR_GST_TAX_ID
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_GST_TAX_ID';
  legacy_value := spraddr_rec.convert_gst_tax_id;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_GST_TAX_ID;
  spraddr_rec.SPRADDR_GST_TAX_ID := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPRADDR_REVIEWED_IND
---------------------------------*/
BEGIN
  cur_col := 'SPRADDR_REVIEWED_IND';
  legacy_value := spraddr_rec.convert_reviewed_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_REVIEWED_IND;
  spraddr_rec.SPRADDR_REVIEWED_IND := converted_val;
```

```
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPRADDR_REVIEWED_USER
-------------------------------*/
BEGIN
  cur_col := 'SPRADDR_REVIEWED_USER';
  legacy_value := spraddr_rec.convert_reviewed_user;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spraddr_rec.CONVERT_REVIEWED_USER;
  spraddr_rec.SPRADDR_REVIEWED_USER := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------------------------
| Updating Temporary table with converted values
-------------------------------------------------*/
BEGIN
  UPDATE SPRADDR_CVT
    SET
    SPRADDR_PIDM = spraddr_rec.spraddr_pidm,
    SPRADDR_ATYP_CODE = spraddr_rec.spraddr_atyp_code,
    SPRADDR_SEQNO = spraddr_rec.spraddr_seqno,
    SPRADDR_FROM_DATE = spraddr_rec.spraddr_from_date,
    SPRADDR_TO_DATE = spraddr_rec.spraddr_to_date,
    SPRADDR_STREET_LINE1 = spraddr_rec.spraddr_street_line1,
    SPRADDR_STREET_LINE2 = spraddr_rec.spraddr_street_line2,
    SPRADDR_STREET_LINE3 = spraddr_rec.spraddr_street_line3,
    SPRADDR_CITY = spraddr_rec.spraddr_city,
    SPRADDR_STAT_CODE = spraddr_rec.spraddr_stat_code,
    SPRADDR_ZIP = spraddr_rec.spraddr_zip,
    SPRADDR_CNTY_CODE = spraddr_rec.spraddr_cnty_code,
    SPRADDR_NATN_CODE = spraddr_rec.spraddr_natn_code,
    SPRADDR_PHONE_AREA = spraddr_rec.spraddr_phone_area,
    SPRADDR_PHONE_NUMBER = spraddr_rec.spraddr_phone_number,
    SPRADDR_PHONE_EXT = spraddr_rec.spraddr_phone_ext,
    SPRADDR_STATUS_IND = spraddr_rec.spraddr_status_ind,
    SPRADDR_ACTIVITY_DATE = SYSDATE,
    SPRADDR_USER = spraddr_rec.spraddr_user,
    SPRADDR_ASRC_CODE = spraddr_rec.spraddr_asrc_code,
    SPRADDR_DELIVERY_POINT = spraddr_rec.spraddr_delivery_point,
```

```
      SPRADDR_CORRECTION_DIGIT = spraddr_rec.spraddr_correction_digit,
      SPRADDR_CARRIER_ROUTE = spraddr_rec.spraddr_carrier_route,
      SPRADDR_GST_TAX_ID = spraddr_rec.spraddr_gst_tax_id,
      SPRADDR_REVIEWED_IND = spraddr_rec.spraddr_reviewed_ind,
      SPRADDR_REVIEWED_USER = spraddr_rec.spraddr_reviewed_user,
      SPRADDR_CVT_STATUS = col_status,
      SPRADDR_CVT_JOB_ID = cur_jobid
      WHERE SPRADDR_CVT_RECORD_ID = cur_rec;
      rows_success_convert := rows_success_convert + 1;
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Error updating record';
      RAISE;
  END;
  <<insert_spraddr>>
  IF process_level = 'B'
  OR process_level = 'I' THEN
    /*---------------------------------------
    |Inserting into Banner Table spraddr
    ----------------------------------------*/
    BEGIN
      INSERT INTO SATURN.spraddr
      (
      SPRADDR_PIDM
      ,SPRADDR_ATYP_CODE
      ,SPRADDR_SEQNO
      ,SPRADDR_FROM_DATE
      ,SPRADDR_TO_DATE
      ,SPRADDR_STREET_LINE1
      ,SPRADDR_CITY
      ,SPRADDR_STAT_CODE
      ,SPRADDR_ZIP
      ,SPRADDR_ACTIVITY_DATE
      )
      VALUES(
      spraddr_rec.SPRADDR_PIDM
      ,spraddr_rec.SPRADDR_ATYP_CODE
      ,spraddr_rec.SPRADDR_SEQNO
      ,spraddr_rec.SPRADDR_FROM_DATE
      ,spraddr_rec.SPRADDR_TO_DATE
      ,spraddr_rec.SPRADDR_STREET_LINE1
      ,spraddr_rec.SPRADDR_CITY
      ,spraddr_rec.SPRADDR_STAT_CODE
      ,spraddr_rec.SPRADDR_ZIP
      ,spraddr_rec.SPRADDR_ACTIVITY_DATE
      );
      rows_success_insert := rows_success_insert + 1;
      UPDATE SPRADDR_CVT
        SET spraddr_cvt_status = 'I'
      WHERE SPRADDR_CVT_RECORD_ID = cur_rec;
    EXCEPTION
      WHEN OTHERS THEN
        err_msg := 'Error inserting into Banner table';
        col_status := 'E';
        cur_col := NULL;
        RAISE;
    END;
  END IF;
  /*---------------------------------------
```

```
    Setting breakpoint values
    ----------------------------------------*/
    break1_value := spraddr_rec.CONVERT_PIDM;
    break2_value := spraddr_rec.CONVERT_ATYP_CODE;
  EXCEPTION
    WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
    WHEN OTHERS THEN
      err_num := SQLCODE;
      err_msg := SUBSTR(SQLERRM, 1, 100);
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      null,null,null,err_msg);
    END;
  IF col_status = 'E' THEN
    rows_errored := rows_errored + 1;
  END IF;
  END LOOP spraddr_loop;
DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPRADDR_CVT:
'||rows_success_convert);
DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spraddr: '||rows_success_insert);
DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
<<end_of_program>>
DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spraddr');
EXCEPTION
WHEN OTHERS THEN
  err_num := SQLCODE;
  err_msg := SUBSTR(SQLERRM, 1, 100);
  process_status := 'E';
  p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
  null,null,null,err_msg);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPRADDR_CVT:
'||rows_success_convert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spraddr: '||rows_success_insert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
  DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spraddr');
END;
/
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') STOP_TIME FROM DUAL;
COMMIT;
PROMPT Querying CURCERR table for errors
col msg for a35 hea 'ERROR MESSAGE' WORD
col colname for a30 hea 'COLUMN_NAME'
col jobno for 99999 hea 'JOB'
col norecs for 999999 hea 'COUNT'
set wrap on
select distinct curcerr_message msg,
       curcerr_column_name colname,
       curcerr_cvt_identifier jobno,
       count(*) norecs
from curcerr
where curcerr_table_name = 'SPRADDR'
group by curcerr_message,
curcerr_column_name,
curcerr_cvt_identifier;
```

```
SPOOL OFF
UNDEFINE process_level
UNDEFINE records_in
```

SPBPERS: General Person Biographic Demographic Table

spbpers_convert.sql

```
/*
=================================================
Process: spbpers_convert.sql
Generated: 30-Sep-2002 08:37
=================================================
AUDIT TRAIL
Converter Tool:  2.10                    SCTCVT   09/30/2002
  This script translates legacy values into Banner values and
  inserts Banner values into spbpers.  This script was
  generated by the SCT Converter Tool.

AUDIT TRAIL END
*/
COLUMN file_id NEW_VALUE spool_file NOPRINT;
SET VERIFY OFF;
SET ECHO OFF;
SELECT 'spbpers_'||cubcnvt_sequence.NEXTVAL||'.log' file_id FROM DUAL;
SPOOL &spool_file;
PROMPT
PROMPT Please enter the record type which should be processed.
PROMPT  (N)ew records
PROMPT  (C)onverted records
PROMPT  (E)rrored records
ACCEPT records_in CHAR PROMPT 'Include:'
PROMPT
PROMPT Please enter the disposition to which the records should
PROMPT be processed.
PROMPT  (C)onvert records only
PROMPT  (I)nsert records only (records were previously converted successfully).
PROMPT  (B)oth convert and insert records into spbpers
ACCEPT process_level CHAR PROMPT 'Disposition: '
PROMPT
SET SERVEROUTPUT ON SIZE 500000;
PROMPT
PROMPT ================================================
PROMPT = will be creating generated id records       =
PROMPT = in SPRIDEN_CVT for any converted records    =
PROMPT =  (i.e. records with spriden_cvt_status = 'C' =
PROMPT ================================================
PROMPT
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') START_TIME FROM DUAL;
DECLARE
  cur_jobid NUMBER := 0;
  cur_owner VARCHAR2(30) := 'SATURN';
  cur_tbl VARCHAR2(30) := 'SPBPERS';
  cur_fileid VARCHAR2(2000);
  col_status  VARCHAR2(1);
  legacy_value VARCHAR2(2000);
  process_status VARCHAR2(1) := 'C';
  err_num  NUMBER(5);
```

```
  col_err VARCHAR2(100);
  cur_rec  NUMBER(8);
  cur_col VARCHAR2(30);
  rows_errored NUMBER(8) := 0;
  rows_success_convert NUMBER(8) := 0;
  rows_success_insert NUMBER(8) := 0;
  commit_counter NUMBER(8);
  commit_frequency NUMBER(8):= 250;
  converted_val VARCHAR2(2000);
  err_msg  VARCHAR2(300);
  first_row BOOLEAN := TRUE;
  column_error EXCEPTION;
  process_level VARCHAR2(1) := UPPER('&process_level');
  records_in VARCHAR2(1) := UPPER('&records_in');
  break1_value VARCHAR2(30);
  break2_value VARCHAR2(30);
  break3_value VARCHAR2(30);
  seq1 NUMBER(5) := 0;
  seq2 NUMBER(5) := 0;
  seq3 NUMBER(5) := 0;
  CURSOR spbpers_cursor IS
    SELECT * FROM spbpers_CVT
     WHERE spbpers_cvt_status = records_in

     order by spbpers_cvt_record_id;
BEGIN
  IF process_level = 'I' AND
     records_in <> 'C' THEN
     DBMS_OUTPUT.PUT_LINE('If inserting only, you must choose to include converted records
only.');
     GOTO end_of_program;
  END IF;
  SELECT cubcnvt_sequence.CURRVAL INTO cur_jobid FROM DUAL;
  DBMS_OUTPUT.ENABLE(100000);
  DBMS_OUTPUT.PUT_LINE('Beginning Conversion of Table spbpers');
  DBMS_OUTPUT.PUT_LINE('Job Number: '||cur_jobid);
  commit_counter := 0;
  <<spbpers_loop>>
  FOR spbpers_rec IN spbpers_cursor LOOP
  BEGIN
     cur_rec := spbpers_rec.spbpers_cvt_record_id;
     col_status := 'C';
     DELETE FROM curcerr
      WHERE curcerr_table_owner = cur_owner
        AND curcerr_table_name = cur_tbl
        AND curcerr_record_id = cur_rec;

     IF process_level = 'I' THEN
        GOTO insert_spbpers;
     END IF;
     /*-------------------------------
     |Beginning evaluation of column
     |SPBPERS_PIDM
     ---------------------------------*/
     BEGIN
        cur_col := 'SPBPERS_PIDM';
        legacy_value := spbpers_rec.convert_pidm;
        /* CURCNVT Rules:
```

```
    Column is Required
    Conversion function is specified
    No validation specified
    No default value
   */
   converted_val := null;
   converted_val := F_CVT_GET_PIDM(LEGACY_VALUE);
   IF SUBSTR(converted_val,1,3) = 'ERR' OR
      SUBSTR(converted_val,1,3) = 'ORA' THEN
      err_msg := 'Column conv failure:'||converted_val;
      RAISE column_error;
   END IF;
   IF converted_val IS NULL THEN
        err_msg := 'Missing required value';
        RAISE column_error;
   END IF;
   BEGIN
     spbpers_rec.SPBPERS_PIDM := TO_NUMBER(converted_val
     );
   EXCEPTION
     WHEN OTHERS THEN
       err_msg := 'Invalid Data Format';
       RAISE column_error;
   END;
EXCEPTION
   WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_SSN
---------------------------------*/
BEGIN
   cur_col := 'SPBPERS_SSN';
   legacy_value := spbpers_rec.convert_ssn;
   /* CURCNVT Rules:
     Column is Not Required
     Conversion function is not specified
     No validation specified
     No default value
   */
   converted_val := null;
   converted_val := spbpers_rec.CONVERT_SSN;
   spbpers_rec.SPBPERS_SSN := converted_val;
EXCEPTION
   WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_BIRTH_DATE
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPBPERS_BIRTH_DATE';
  legacy_value := spbpers_rec.convert_birth_date;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_BIRTH_DATE;

--THIS SECTION ILLUSTRATES USING THE DATE FORMAT MASK
    BEGIN
      spbpers_rec.SPBPERS_BIRTH_DATE := TO_DATE(converted_val
      ,'YYYYMMDD'
      ); -- FORMATTING FOR DATES
    EXCEPTION
      WHEN OTHERS THEN
        err_msg := 'Invalid Data Format';
        RAISE column_error;
    END;
  EXCEPTION
    WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
  END;
  /*-------------------------------
  |Beginning evaluation of column
  |SPBPERS_LGCY_CODE
  ---------------------------------*/
  BEGIN
    cur_col := 'SPBPERS_LGCY_CODE';
    legacy_value := spbpers_rec.convert_lgcy_code;
    /* CURCNVT Rules:
      Column is Not Required
      Conversion function is not specified
      No validation specified
      No default value
    */
    converted_val := null;
    converted_val := spbpers_rec.CONVERT_LGCY_CODE;
    spbpers_rec.SPBPERS_LGCY_CODE := converted_val;
  EXCEPTION
    WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
  END;
  /*-------------------------------
  |Beginning evaluation of column
  |SPBPERS_MRTL_CODE
  ---------------------------------*/
```

```
BEGIN
  cur_col := 'SPBPERS_MRTL_CODE';
  legacy_value := spbpers_rec.convert_mrtl_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    Validation function specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_MRTL_CODE;
  spbpers_rec.SPBPERS_MRTL_CODE := converted_val;
  col_err :=

--THIS SECTION ILLUSTRATES USING THE VALIDATION FUNCTION
F_VALIDATE_SINGLE('SATURN','STVMRTL','STVMRTL_CODE',LEGACY_VALUE);
    IF col_err = 'TRUE' THEN
      err_msg := 'Validation failure';
      RAISE column_error;  ---USING F_VALIDATE_SINGLE FUNCTION
    ELSIF SUBSTR(col_err,1,3) = 'ERR' THEN
      err_msg := col_err;
      RAISE column_error;
    END IF;
  EXCEPTION
    WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
  END;
  /*-------------------------------
  |Beginning evaluation of column
  |SPBPERS_ETHN_CODE
  ---------------------------------*/
  BEGIN
    cur_col := 'SPBPERS_ETHN_CODE';
    legacy_value := spbpers_rec.convert_ethn_code;
    /* CURCNVT Rules:
      Column is Not Required
      Conversion function is specified
      No validation specified
      No default value

      */ THIS SECTION ILLUSTRATES THE USE OF THE CROSSWALK FUNCTION
      converted_val := null;
      converted_val := F_CVT_CURCVAL_RNULL('STVETHN',LEGACY_VALUE);
      IF SUBSTR(converted_val,1,3) = 'ERR' OR
        SUBSTR(converted_val,1,3) = 'ORA' THEN
        err_msg := 'Column conv failure:'||converted_val;
        RAISE column_error;
      END IF; ---USING CROSSWALK FUNCTION
    spbpers_rec.SPBPERS_ETHN_CODE := converted_val;
  EXCEPTION
    WHEN column_error THEN
      col_status := 'E';
      process_status := 'E';
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
```

```
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_RELG_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_RELG_CODE';
  legacy_value := spbpers_rec.convert_relg_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_RELG_CODE;
  spbpers_rec.SPBPERS_RELG_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_SEX
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_SEX';
  legacy_value := spbpers_rec.convert_sex;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
   Validation list specified
    No default value
  */
    converted_val := null;
    converted_val := spbpers_rec.CONVERT_SEX;
    spbpers_rec.SPBPERS_SEX := converted_val;

--THIS SECTION ILLUSTRATES THE USER OF THE LIST OF VALUES

    IF spbpers_rec.SPBPERS_SEX NOT IN ('M','F','N') THEN
      err_msg := 'Validation failure';
      RAISE column_error;  --- USING LIST OF VALUES
  END IF;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_CONFID_IND
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPBPERS_CONFID_IND';
  legacy_value := spbpers_rec.convert_confid_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_CONFID_IND;
  spbpers_rec.SPBPERS_CONFID_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPBPERS_DEAD_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_DEAD_IND';
  legacy_value := spbpers_rec.convert_dead_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_DEAD_IND;
  spbpers_rec.SPBPERS_DEAD_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPBPERS_VETC_FILE_NUMBER
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_VETC_FILE_NUMBER';
  legacy_value := spbpers_rec.convert_vetc_file_number;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_VETC_FILE_NUMBER;
  spbpers_rec.SPBPERS_VETC_FILE_NUMBER := converted_val;
```

```
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_LEGAL_NAME
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_LEGAL_NAME';
  legacy_value := spbpers_rec.convert_legal_name;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_LEGAL_NAME;
  spbpers_rec.SPBPERS_LEGAL_NAME := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_PREF_FIRST_NAME
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_PREF_FIRST_NAME';
  legacy_value := spbpers_rec.convert_pref_first_name;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_PREF_FIRST_NAME;
  spbpers_rec.SPBPERS_PREF_FIRST_NAME := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_NAME_PREFIX
---------------------------------*/
```

```
BEGIN
  cur_col := 'SPBPERS_NAME_PREFIX';
  legacy_value := spbpers_rec.convert_name_prefix;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_NAME_PREFIX;
  spbpers_rec.SPBPERS_NAME_PREFIX := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPBPERS_NAME_SUFFIX
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_NAME_SUFFIX';
  legacy_value := spbpers_rec.convert_name_suffix;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_NAME_SUFFIX;
  spbpers_rec.SPBPERS_NAME_SUFFIX := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*---------------------------------
|Beginning evaluation of column
|SPBPERS_ACTIVITY_DATE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_ACTIVITY_DATE';
  legacy_value := spbpers_rec.convert_activity_date;
  /* CURCNVT Rules:
    Column is Required
    Conversion function is specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := SYSDATE;
```

```
  IF SUBSTR(converted_val,1,3) = 'ERR' OR
    SUBSTR(converted_val,1,3) = 'ORA' THEN
    err_msg := 'Column conv failure:'||converted_val;
    RAISE column_error;
  END IF;
  IF converted_val IS NULL THEN
      err_msg := 'Missing required value';
      RAISE column_error;
  END IF;
  BEGIN
    spbpers_rec.SPBPERS_ACTIVITY_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_VERA_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_VERA_IND';
  legacy_value := spbpers_rec.convert_vera_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_VERA_IND;
  spbpers_rec.SPBPERS_VERA_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_CITZ_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_CITZ_IND';
  legacy_value := spbpers_rec.convert_citz_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
```

```
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_CITZ_IND;
  spbpers_rec.SPBPERS_CITZ_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_DEAD_DATE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_DEAD_DATE';
  legacy_value := spbpers_rec.convert_dead_date;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_DEAD_DATE;
  BEGIN
    spbpers_rec.SPBPERS_DEAD_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_PIN
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_PIN';
  legacy_value := spbpers_rec.convert_pin;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
```

```
 converted_val := spbpers_rec.CONVERT_PIN;
  spbpers_rec.SPBPERS_PIN := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPBPERS_CITZ_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_CITZ_CODE';
  legacy_value := spbpers_rec.convert_citz_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_CITZ_CODE;
  spbpers_rec.SPBPERS_CITZ_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
|Beginning evaluation of column
|SPBPERS_HAIR_CODE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_HAIR_CODE';
  legacy_value := spbpers_rec.convert_hair_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_HAIR_CODE;
  spbpers_rec.SPBPERS_HAIR_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*--------------------------------
```

```
|Beginning evaluation of column
|SPBPERS_EYES_CODE
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_EYES_CODE';
  legacy_value := spbpers_rec.convert_eyes_code;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_EYES_CODE;
  spbpers_rec.SPBPERS_EYES_CODE := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_CITY_BIRTH
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_CITY_BIRTH';
  legacy_value := spbpers_rec.convert_city_birth;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_CITY_BIRTH;
  spbpers_rec.SPBPERS_CITY_BIRTH := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_STAT_CODE_BIRTH
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_STAT_CODE_BIRTH';
  legacy_value := spbpers_rec.convert_stat_code_birth;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
```

```
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_STAT_CODE_BIRTH;
  spbpers_rec.SPBPERS_STAT_CODE_BIRTH := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_DRIVER_LICENSE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_DRIVER_LICENSE';
  legacy_value := spbpers_rec.convert_driver_license;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_DRIVER_LICENSE;
  spbpers_rec.SPBPERS_DRIVER_LICENSE := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_STAT_CODE_DRIVER
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_STAT_CODE_DRIVER';
  legacy_value := spbpers_rec.convert_stat_code_driver;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_STAT_CODE_DRIVER;
  spbpers_rec.SPBPERS_STAT_CODE_DRIVER := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
```

```
|Beginning evaluation of column
|SPBPERS_NATN_CODE_DRIVER
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_NATN_CODE_DRIVER';
  legacy_value := spbpers_rec.convert_natn_code_driver;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_NATN_CODE_DRIVER;
  spbpers_rec.SPBPERS_NATN_CODE_DRIVER := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_UOMS_CODE_HEIGHT
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_UOMS_CODE_HEIGHT';
  legacy_value := spbpers_rec.convert_uoms_code_height;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_UOMS_CODE_HEIGHT;
  spbpers_rec.SPBPERS_UOMS_CODE_HEIGHT := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_HEIGHT
----------------------------------*/
BEGIN
  cur_col := 'SPBPERS_HEIGHT';
  legacy_value := spbpers_rec.convert_height;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
```

```
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_HEIGHT;
  BEGIN
    spbpers_rec.SPBPERS_HEIGHT := TO_NUMBER(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_UOMS_CODE_WEIGHT
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_UOMS_CODE_WEIGHT';
  legacy_value := spbpers_rec.convert_uoms_code_weight;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_UOMS_CODE_WEIGHT;
  spbpers_rec.SPBPERS_UOMS_CODE_WEIGHT := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_WEIGHT
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_WEIGHT';
  legacy_value := spbpers_rec.convert_weight;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_WEIGHT;
  BEGIN
    spbpers_rec.SPBPERS_WEIGHT := TO_NUMBER(converted_val
    );
```

```
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_SDVET_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_SDVET_IND';
  legacy_value := spbpers_rec.convert_sdvet_ind;
  /* CURCNVT Rules:
   Column is Not Required
   Conversion function is not specified
   No validation specified
   No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_SDVET_IND;
  spbpers_rec.SPBPERS_SDVET_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_LICENSE_ISSUED_DATE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_LICENSE_ISSUED_DATE';
  legacy_value := spbpers_rec.convert_license_issued_date;
  /* CURCNVT Rules:
   Column is Not Required
   Conversion function is not specified
   No validation specified
   No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_LICENSE_ISSUED_DATE;
  BEGIN
    spbpers_rec.SPBPERS_LICENSE_ISSUED_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
```

```
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_LICENSE_EXPIRES_DATE
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_LICENSE_EXPIRES_DATE';
  legacy_value := spbpers_rec.convert_license_expires_date;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_LICENSE_EXPIRES_DATE;
  BEGIN
    spbpers_rec.SPBPERS_LICENSE_EXPIRES_DATE := TO_DATE(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_INCAR_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_INCAR_IND';
  legacy_value := spbpers_rec.convert_incar_ind;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_INCAR_IND;
  spbpers_rec.SPBPERS_INCAR_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
```

```
      p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
      legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_WEBID
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_WEBID';
  legacy_value := spbpers_rec.convert_webid;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_WEBID;
  spbpers_rec.SPBPERS_WEBID := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_WEB_LAST_ACCESS
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_WEB_LAST_ACCESS';
  legacy_value := spbpers_rec.convert_web_last_access;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_WEB_LAST_ACCESS;
  spbpers_rec.SPBPERS_WEB_LAST_ACCESS := converted_val;
EXCEPTION
  WHEN column_error THEN
     col_status := 'E';
     process_status := 'E';
     p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
     legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_PIN_DISABLED_IND
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_PIN_DISABLED_IND';
  legacy_value := spbpers_rec.convert_pin_disabled_ind;
  /* CURCNVT Rules:
```

```
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_PIN_DISABLED_IND;
  spbpers_rec.SPBPERS_PIN_DISABLED_IND := converted_val;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------
|Beginning evaluation of column
|SPBPERS_ITIN
---------------------------------*/
BEGIN
  cur_col := 'SPBPERS_ITIN';
  legacy_value := spbpers_rec.convert_itin;
  /* CURCNVT Rules:
    Column is Not Required
    Conversion function is not specified
    No validation specified
    No default value
  */
  converted_val := null;
  converted_val := spbpers_rec.CONVERT_ITIN;
  BEGIN
    spbpers_rec.SPBPERS_ITIN := TO_NUMBER(converted_val
    );
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Invalid Data Format';
      RAISE column_error;
  END;
EXCEPTION
  WHEN column_error THEN
    col_status := 'E';
    process_status := 'E';
    p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
    legacy_value,converted_val,null,err_msg);
END;
/*-------------------------------------------------
| Updating Temporary table with converted values
-------------------------------------------------*/
BEGIN
  UPDATE SPBPERS_CVT
    SET
    SPBPERS_PIDM = spbpers_rec.spbpers_pidm,
    SPBPERS_SSN = spbpers_rec.spbpers_ssn,
    SPBPERS_BIRTH_DATE = spbpers_rec.spbpers_birth_date,
    SPBPERS_LGCY_CODE = spbpers_rec.spbpers_lgcy_code,
    SPBPERS_MRTL_CODE = spbpers_rec.spbpers_mrtl_code,
    SPBPERS_ETHN_CODE = spbpers_rec.spbpers_ethn_code,
```

```
      SPBPERS_RELG_CODE = spbpers_rec.spbpers_relg_code,
      SPBPERS_SEX = spbpers_rec.spbpers_sex,
      SPBPERS_CONFID_IND = spbpers_rec.spbpers_confid_ind,
      SPBPERS_DEAD_IND = spbpers_rec.spbpers_dead_ind,
      SPBPERS_VETC_FILE_NUMBER = spbpers_rec.spbpers_vetc_file_number,
      SPBPERS_LEGAL_NAME = spbpers_rec.spbpers_legal_name,
      SPBPERS_PREF_FIRST_NAME = spbpers_rec.spbpers_pref_first_name,
      SPBPERS_NAME_PREFIX = spbpers_rec.spbpers_name_prefix,
      SPBPERS_NAME_SUFFIX = spbpers_rec.spbpers_name_suffix,
      SPBPERS_ACTIVITY_DATE = SYSDATE,
      SPBPERS_VERA_IND = spbpers_rec.spbpers_vera_ind,
      SPBPERS_CITZ_IND = spbpers_rec.spbpers_citz_ind,
      SPBPERS_DEAD_DATE = spbpers_rec.spbpers_dead_date,
      SPBPERS_PIN = spbpers_rec.spbpers_pin,
      SPBPERS_CITZ_CODE = spbpers_rec.spbpers_citz_code,
      SPBPERS_HAIR_CODE = spbpers_rec.spbpers_hair_code,
      SPBPERS_EYES_CODE = spbpers_rec.spbpers_eyes_code,
      SPBPERS_CITY_BIRTH = spbpers_rec.spbpers_city_birth,
      SPBPERS_STAT_CODE_BIRTH = spbpers_rec.spbpers_stat_code_birth,
      SPBPERS_DRIVER_LICENSE = spbpers_rec.spbpers_driver_license,
      SPBPERS_STAT_CODE_DRIVER = spbpers_rec.spbpers_stat_code_driver,
      SPBPERS_NATN_CODE_DRIVER = spbpers_rec.spbpers_natn_code_driver,
      SPBPERS_UOMS_CODE_HEIGHT = spbpers_rec.spbpers_uoms_code_height,
      SPBPERS_HEIGHT = spbpers_rec.spbpers_height,
      SPBPERS_UOMS_CODE_WEIGHT = spbpers_rec.spbpers_uoms_code_weight,
      SPBPERS_WEIGHT = spbpers_rec.spbpers_weight,
      SPBPERS_SDVET_IND = spbpers_rec.spbpers_sdvet_ind,
      SPBPERS_LICENSE_ISSUED_DATE = spbpers_rec.spbpers_license_issued_date,
      SPBPERS_LICENSE_EXPIRES_DATE = spbpers_rec.spbpers_license_expires_date,
      SPBPERS_INCAR_IND = spbpers_rec.spbpers_incar_ind,
      SPBPERS_WEBID = spbpers_rec.spbpers_webid,
      SPBPERS_WEB_LAST_ACCESS = spbpers_rec.spbpers_web_last_access,
      SPBPERS_PIN_DISABLED_IND = spbpers_rec.spbpers_pin_disabled_ind,
      SPBPERS_ITIN = spbpers_rec.spbpers_itin,
      SPBPERS_CVT_STATUS = col_status,
      SPBPERS_CVT_JOB_ID = cur_jobid
      WHERE SPBPERS_CVT_RECORD_ID = cur_rec;
      rows_success_convert := rows_success_convert + 1;
  EXCEPTION
    WHEN OTHERS THEN
      err_msg := 'Error updating record';
      RAISE;
  END;
  <<insert_spbpers>>
  IF process_level = 'B'
  OR process_level = 'I' THEN
    /*----------------------------------------
    |Inserting into Banner Table spbpers
    ----------------------------------------*/
    BEGIN
      INSERT INTO SATURN.spbpers
      (
      SPBPERS_PIDM
      ,SPBPERS_SSN
      ,SPBPERS_BIRTH_DATE
      ,SPBPERS_MRTL_CODE
      ,SPBPERS_ETHN_CODE
```

```
            ,SPBPERS_SEX
            ,SPBPERS_ACTIVITY_DATE
            )
            VALUES(
            spbpers_rec.SPBPERS_PIDM
            ,spbpers_rec.SPBPERS_SSN
            ,spbpers_rec.SPBPERS_BIRTH_DATE
            ,spbpers_rec.SPBPERS_MRTL_CODE
            ,spbpers_rec.SPBPERS_ETHN_CODE
            ,spbpers_rec.SPBPERS_SEX
            ,spbpers_rec.SPBPERS_ACTIVITY_DATE
            );
            rows_success_insert := rows_success_insert + 1;
            UPDATE SPBPERS_CVT
              SET spbpers_cvt_status = 'I'
            WHERE SPBPERS_CVT_RECORD_ID = cur_rec;
          EXCEPTION
            WHEN OTHERS THEN
              err_msg := 'Error inserting into Banner table';
              col_status := 'E';
              cur_col := NULL;
              RAISE;
          END;
        END IF;
      EXCEPTION
        WHEN column_error THEN
          col_status := 'E';
          process_status := 'E';
          p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
          legacy_value,converted_val,null,err_msg);
        WHEN OTHERS THEN
          err_num := SQLCODE;
          err_msg := SUBSTR(SQLERRM, 1, 100);
          process_status := 'E';
          p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
          null,null,null,err_msg);
        END;
      IF col_status = 'E' THEN
        rows_errored := rows_errored + 1;
      END IF;
      if commit_counter = commit_frequency then
        commit;
        commit_counter := 0;
      else
        commit_counter := commit_counter + 1;
      end if;
      END LOOP spbpers_loop;
    DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPBPERS_CVT:
    '||rows_success_convert);
    DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spbpers: '||rows_success_insert);
    DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
    <<end_of_program>>
    DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spbpers');
    EXCEPTION
    WHEN OTHERS THEN
      err_num := SQLCODE;
      err_msg := SUBSTR(SQLERRM, 1, 100);
```

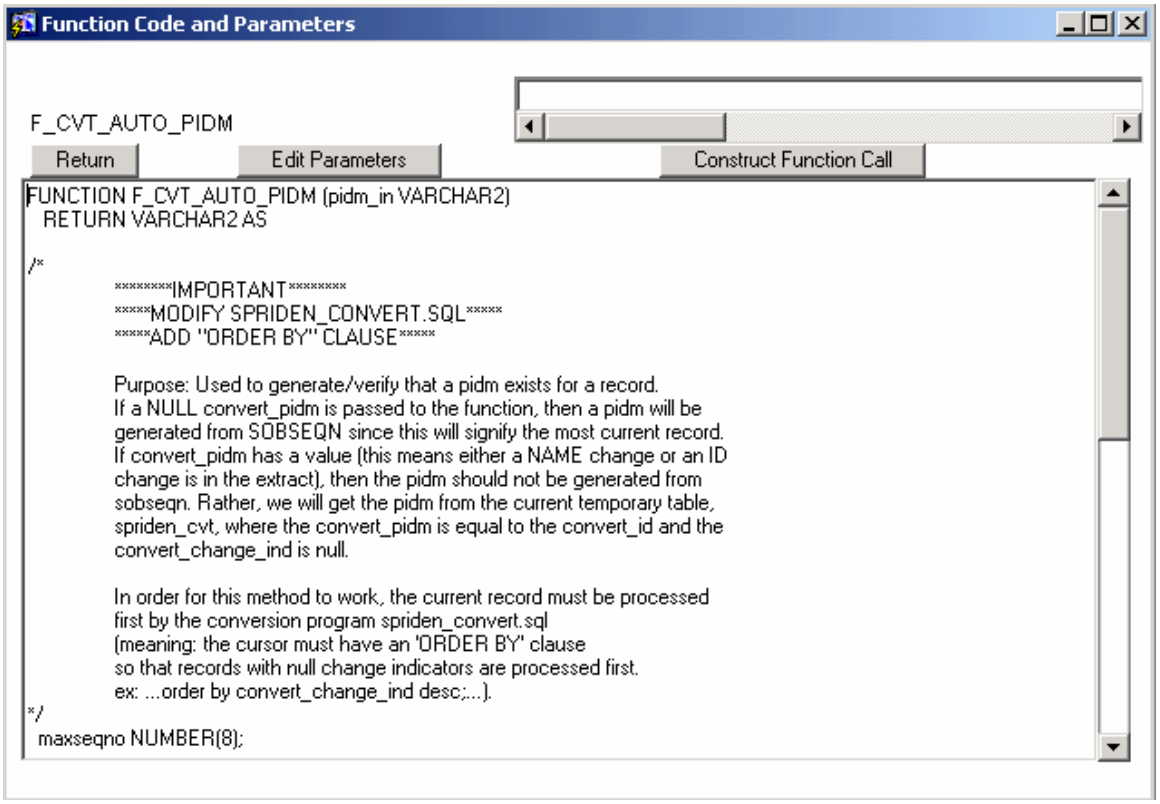
```
  process_status := 'E';
  p_cvt_err(cur_owner,cur_tbl, cur_jobid, cur_rec, cur_col,
  null,null,null,err_msg);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Converted in SPBPERS_CVT:
'||rows_success_convert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows Inserted into spbpers: '||rows_success_insert);
  DBMS_OUTPUT.PUT_LINE('Number of Rows That Errored: '||rows_errored);
  DBMS_OUTPUT.PUT_LINE(CHR(10)||'Completed Processing of spbpers');
END;
/
SELECT TO_CHAR(sysdate, 'DD-MON-YYYY HH24:MI') STOP_TIME FROM DUAL;
COMMIT;
PROMPT Querying CURCERR table for errors
col msg for a35 hea 'ERROR MESSAGE' WORD
col colname for a30 hea 'COLUMN_NAME'
col jobno for 99999 hea 'JOB'
col norecs for 999999 hea 'COUNT'
set wrap on
select distinct curcerr_message msg,
      curcerr_column_name colname,
      curcerr_cvt_identifier jobno,
      count(*) norecs
from curcerr
where curcerr_table_name = 'SPBPERS'
group by curcerr_message,
curcerr_column_name,
curcerr_cvt_identifier;
SPOOL OFF
UNDEFINE process_level
UNDEFINE records_in
```

Appendix E - SCT Converter Tool Functions

Viewing Functions E-3
Editing Functions Online E-4

Viewing Functions

The database functions provided with the Converter Tool can be helpful in getting data from your legacy system to SCT Banner. The source code and internal documentation for the functions delivered with the Converter Tool can be viewed in the Converter Tool install directory in the cvt_genlib.sql file on the database server. You can also view a function source code by placing the cursor on the field that contains the function name and pressing the VIEW FUNCTION button.



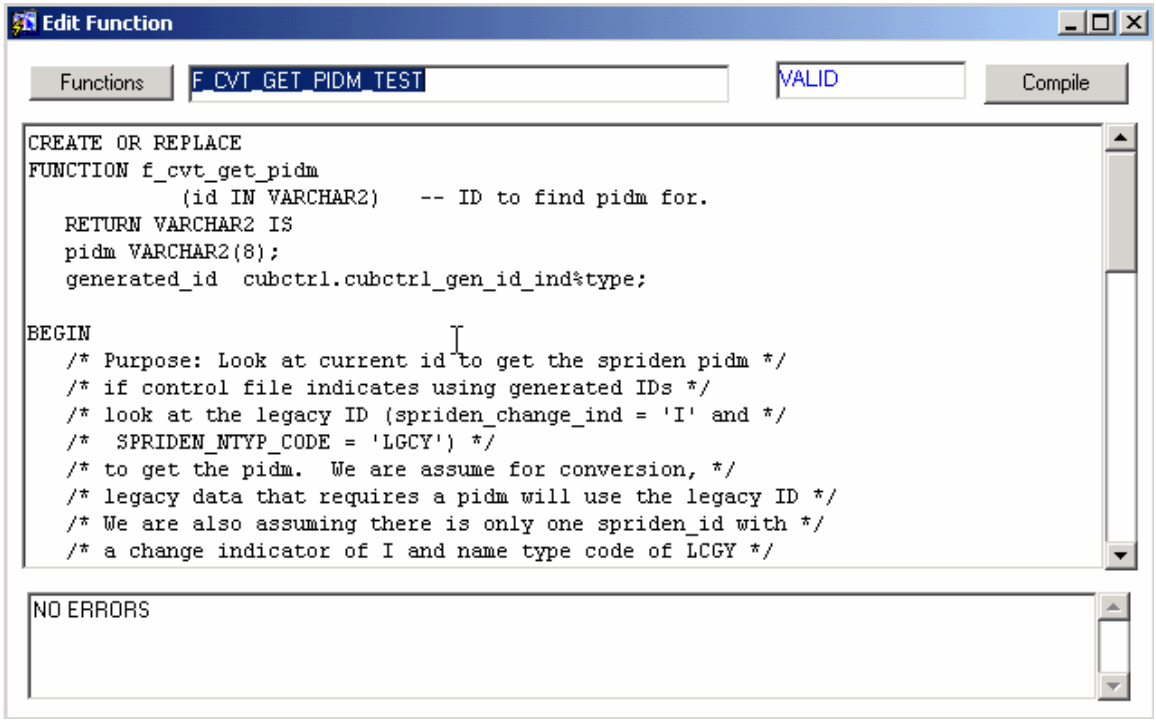
Editing Functions Online

You may wish to create new functions to add to your database as you work through a conversion process. You may add to your function library in two ways. First, you may choose to write the function script and compile it in the database instance in which you are working (logged in as your conversion user). The next time you open CUACNVT and look at functions, your new function should appear on the list.

Or you may choose to edit an existing function online. To do this, click the Edit Function button on the Toolbar:

This opens the Edit Function window, where you can edit an existing function, rename it, compile it, and add it to your function library.

This image demonstrates the procedure using the delivered F_CVT_GET_PIDM function. It has been edited, given a new name, F_CVT_GET_PIDM_TEST, and recompiled under the new name. This operates the same way as a "save as" feature in a text processing application, in that the original object is preserved and a new one created with a different name and different properties.



For more discussion on database functions and the converter tool refer to The Function Code and Parameters Window in Chapter 2.

Appendix F - Using the SCT Converter Tool

SCT Converter Tool Instructions F-3
Using CUACNVT to produce the <TargetTable>_cvt_create.sql script F-7
Using CUACNVT to produce the <TargetTable>_cvt.ctl script F-8
Using CUACNVT to produce the <TargetTable>_convert.sql script F-9

SCT Converter Tool Instructions

The following steps will show how to use the CUACNVT form to set up specifications for any conversion.

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Open CUACNVT and log in. Login: your_convert_username Password: your_convert_user_password Database: your_database	CUACNVT	
Click on the Control Form icon on the Toolbar.		Control form is displayed
Select "Use Generated ID's" if you plan to do so. Identify your default directory for output. SAVE your changes and EXIT the form.		CUACNVT form displays.
Insert a new record	CUACNVT Table Block	Record/Insert on the Menu OR Press the F6 key OR Use the down arrow key to Navigate to a blank record.
Enter the name of the table owner		
Navigate to the Next Record.		
Enter the name of the target table.		
Navigate to the "Error Action" field.		Accept the default Error Action, "Continue Processing Row."
Navigate to the Data Input Format field and select the data format and delimiter if appropriate.		
SAVE your changes		As you save your changes, the column block is populated with all the columns from the target table
NEXT BLOCK	CUACNVT Column Block	To see all columns in the table, after moving to the Column Block, with your cursor in the "Column" field, move through the records with the NEXT RECORD button or by using the down arrow.
The cursor is in the Column field, and the first column in the table is displayed.		Use the PREVIOUS RECORD button or the Up arrow to move back through the records to the first column.
Navigate to the Load Order field if you need to change the order of the columns to be loaded by the SQL*Loader control file.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Navigate to the "Convert Fnctn" field if you wish to apply a function to the selected column.		If you wish to use a function on a field, the following steps outline that process.
Press the CONVERT FNCTN button.		
Choose the appropriate function from the list.		
With your cursor in the "Convert Fnctn" field, press the VIEW FUNCTION button on the Tool Bar.	Function Code and Parameters Window	You will see the "Function Code and Parameters" window. It shows the text of the function create script, and has three buttons across the top of the window above the function text.
Press the EDIT PARAMETERS button.	Edit Parameters Dialog Box	The left side of the parameters dialog box has the parameter names defaulted in.
Enter the correct parameters for the function you are using.		LEGACY_VALUE is a variable that has been defined in the convert script generated by the converter tool. It refers to the legacy data value in the *convert* column in the convert table. Any time it is used as a parameter in a function, the value passed to the function is the legacy data value that resides in the *convert* column corresponding to the target table column to which the function is being applied.
Press the CONSTRUCT FUNCTION CALL button.		When you press the CONSTRUCT FUNCTION CALL button, the blank box at the top right corner of the form is populated with the actual function call that will appear in the convert script.
Press the RETURN button.		
Choose "Save & Return."		Doing this will return your cursor to the Table Block of CUACNVT. To resume work, you must perform a NEXT BLOCK function.
NEXT BLOCK	CUACNVT	Exiting the "Function Code and Parameters" window places you in the Table Block.
Move through all remaining columns, entering the necessary specifications as you go.	CUACNVT Column Block	Use the down arrow or the NEXT RECORD button on the Tool Bar to move through the remaining columns in the target table, entering the necessary specifications as you go. If you enter no specification, the legacy value will be moved from the *convert* column to the <*TargetTable*> column in the convert table and subsequently to the actual <*TargetTable*> column, with no actions taken.
To enter a Default Value for a column: With your cursor in the appropriate record, Navigate to the "Default" field.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Enter the default value in the field.		Do not use any quotation marks.
Navigate to the "Default Action" column.		The "Default Action" field requires a value if an entry is made in the "Default" field.
To enter a date format mask: With your cursor in the appropriate record, Navigate to the "Format Mask" field.		
Enter the date format of the legacy date: MMDDYY, MM/DD/YY, etc. (no quotation marks)		The format entered in the format mask field is the legacy data date field format. The format mask here will allow the convert script to recognize the value in the *convert* column as a date to be converted into the correct system date format.
To use a Value List: With your cursor in the appropriate record, Navigate to the "Value List" field.		
Enter a list of acceptable values in that column. Correct format is ("value1", "value2", "value3", etc.)		
To use a Validation Function: With your cursor in the appropriate record, Navigate to the "Valid Fnctn" field.		
Press the VALID FNCTN button and choose F_VALIDATE_SINGLE from the list.		Use this function to validate codes. Pass in the owner, table, and column of the validation table, along with the value you are verifying. The function will validate the code in the *convert* column against the specified validation table and return an error if the code does not exist in the validation table.
Click the VIEW FUNCTION button on the Tool Bar.		
Click the EDIT PARAMETERS button and enter the following parameters: OWNER = 'TABLE_OWNER' TABLENAME = 'VALIDATION_TABLE_NAME' COLUMN = 'VALIDATION_TABLE_COLUMN' IN_VALUE = LEGACY_VALUE		The parameters for OWNER, TABLENAME, and COLUMN must be enclosed in single quotes. LEGACY_VALUE does not require quotes.
Click the Construct Function Call button		
Click the Return button.		

ACTIONS/STEPS	FORM/PROCESS	NOTES & HINTS
Choose "Save & Return."		You will be returned to the Table Block of the form and must perform a NEXT BLOCK function to return to the Column Block.
Navigate to the "Load" field in the Column Block and make the appropriate choice for the "Load" indicator.		Checking the LOAD button indicates that you want this column to appear in the SQL*Loader script because you have legacy data in the flat file for this column to be loaded into the convert table. UNchecking the Load box indicates that there is no legacy data to be loaded into the column. There may be a default value applied by a function, but the Load box should be checked ONLY if there is legacy data in the data file for this column.
Navigate to the "Insert" indicator in the Column Block and make the appropriate choice for the "Insert" indicator.		Checking the INSERT button indicates that you want this column to appear in the conversion script to be converted to the target table columns in the convert table and subsequently inserted into the actual target table. Not all columns need be inserted. If a column is nullable and if there is no data nor any place holder for data in your legacy data file, you may UNcheck both Load and Insert. If there is data in one record for this column or if there is a function that will apply data to this column, you must check the Insert box.
When all specifications for conversion have been entered, SAVE your changes.		

You have now entered all the necessary specifications for conversion into any target table. The next step is to use the features of the converter tool to produce the following 3 scripts:

- <TargetTable>_cvt_create.sql – creates the convert table (<TargetTable>_cvt)
- <TargetTable>_cvt.ctl – used by SQL*Loader to load the data from the client's flat file (<TargetTable>_cvt.dat) to the convert table
- <TargetTable>_convert.sql – copies the data from the convert columns to the target table columns within the convert table, then moves the data from the target table columns in the convert table to the corresponding columns in the actual target table in the database.

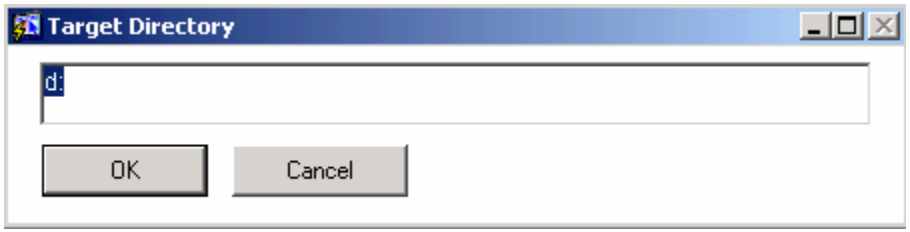
Using CUACNVT to produce the *<TargetTable>_cvt_create.sql* script

With the cursor in the Table Block for the table:

Press the CONVERT TABLE SCRIPT button: 

If you are using Internet Native forms and writing to your database server, no dialog box will appear.

If you are using client/server mode and have chosen "Local Machine" on the Control Form, the following dialog box will appear:



The box will be populated with the directory value you entered on the Control Form.

When the script is complete, you will see the following message on the message line of the form:

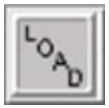
"Script successfully written to c:\convert\<TargetTable>_cvt_create.sql."

To view the text of a sample convert script, refer to Appendix B: Sample Table Creation Scripts.

Run this script on the server to create the convert table in the host database.

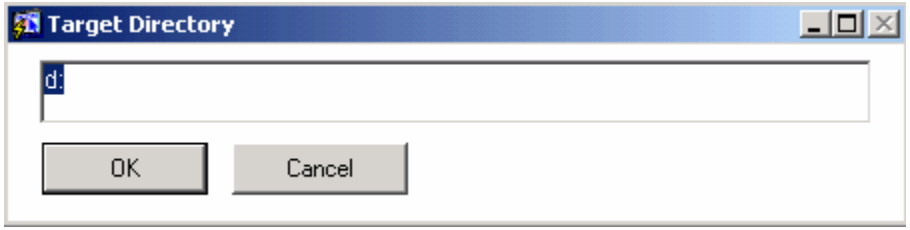
Using CUACNVT to produce the <*TargetTable*>_*cvt.ctl* script

With the cursor in the Table Block for the table:

Press the SQL LOADER SCRIPT button: 

If you are using Internet Native forms and writing to your database server, no dialog box will appear.

If you are using client/server mode and have chosen "Local Machine" on the Control Form, the following dialog box will appear:



The box will be populated with the directory value you entered on the Control Form.

Press "OK."

When the script is complete, you will see the following message on the message line of the form:

"Script successfully written to c:\convert\<TargetTable>_cvt.ctl"

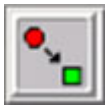
To view the text of the script, refer to Appendix C: Sample SQL*Loader Scripts.

Place this script and the <*TargetTable*>_*cvt.dat* data file in the same directory on the server and run SQL*Loader to load the data into the convert table. Consult with your DBA for the correct command for executing SQL*Loader. A sample command:

sqlldr userid=your_convert_username/your_convert_user_password control=<TargetTable>_cvt.ctl

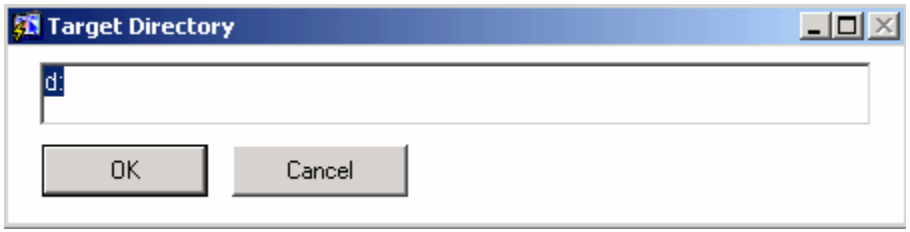
Using CUACNVT to produce the *<TargetTable>_convert.sql* script

With the cursor in the Table Block for the table:

Press the CONVERSION SCRIPT button: 

If you are using Internet Native forms and writing to your database server, no dialog box will appear.

If you are using client/server mode and have chosen "Local Machine" on the Control Form, the following dialog box will appear:



The box will be populated with the directory value you entered on the Control Form.

Press "OK."

When the script is complete, you will see the following message on the message line of the form:

"Script successfully written to c:\convert\<TargetTable>_convert.sql"

To view the text of the script, refer to Appendix D: Sample Conversion Scripts.

Once you have evaluated the data in the convert table *convert* columns, run this script for the (N)ew records in (C)onvert mode. If there are errors, correct them, and run this script again, this time choosing (E)rrored records for the first parameter of the script and (C)onvert for the disposition. Continue this process until all errors are corrected. Once you are satisfied that all errors are correct and that your data is "clean," run this script for the (C)onverted records in (I)nsert mode to insert the data into the target table.

Appendix G – Utility Scripts

\util Directory Scripts G-3
Utility Script Usage G-3
SPRIDEN duplicate checking steps G-3
SPRADDR duplicate checking steps G-4
SPRTELE duplicate checking steps G-4
Extracting Data from the Audit Tables G-5
Validating Data for Crosswalked Columns G-5
Validating Data for Columns with Foreign Key Constraints G-5

\util Directory Scripts

These scripts are delivered with the server.zip file for the converter tool and are provided to help validate data that has been loaded into a temporary table prior to running the convert script. This appendix lists the scripts and how they are used.

- Address_match_report.sql
- Address_type_match_different_street_report.sql
- Create_get_rules.sql
- Cvt_chkfkeys.sql
- Cvt_chkxwlk.sql
- Cvt_status.sql
- Czkrule.sql
- Czkrul1.sql
- Id_duplicate.sql
- Id_last_name_duplicate.sql
- Id_ssn_duplicate.sql
- Spraddr_duplicates.sql
- Spraddr_duplicate_checking_steps.txt
- Spriden_duplicates.sql
- Spriden_duplicate_checking_steps.txt
- Sprtele_duplicate_checking_steps.txt
- Sprtele_duplicates.sql
- Telephone_and_type_match_report.sql

Utility Script Usage

SPRIDEN duplicate checking steps

STEP 1: Run “spriden_duplicates.sql”

This script will check the spriden table to determine if the record in the spriden_cvt table already exists in spriden. It should be run after loading spriden_cvt and spbpers_cvt. The user is prompted for the legacy date format of the convert_birth_date field in the spbpers_cvt table.

If it is determined the id may already exist in SCT Banner, the spriden_cvt_status is set to a specific value which is checked by the following three SQL scripts.

STEP 2: Run “id_duplicate.sql”

This report lists the records in spriden_cvt where convert_id already exists in SPRIDEN but last names do not match.

STEP 3: Run “id_last_name_duplicate.sql”

This report lists the records in the spriden_cvt table where the id and last name already exists in SPRIDEN.

STEP 4: RUN “id_ssn_duplicate.sql”

This report list the records in the spriden_cvt temp table where their convert_id already exists in SPRIDEN and their convert_ssn already exists in SPBPERS. The convert_last_name in the temp table, however, does not match the spriden_last_name in SPRIDEN.

SPRADDR duplicate checking steps

STEP 1: Run “spraddr_duplicates.sql”

This program will check that the legacy address information has not been loaded previously for a specific address type. If a match is found on id(pidm),address type and the first seven characters of street line1... then the SPRADDR_CVT_STATUS is set to “D” and the record will not be loaded since it already exists. If a match is found on id(pidm),address type but the street line1 does not match and both the legacy address and the SCT Banner address are active, then the SPRADDR_CVT_STATUS is set to “X” so a report can be produced for the client to decide which address record should be the active record for the address type (manual updating of either the legacy address or the SCT Banner address to inactive and re-extract of data file should be done once legacy records are accurate).

STEP 2: RUN "address_match_report.sql"

Will produce "address_match_report.lis" which is a listing of the records that will not be loaded since an exact match was found.

STEP 3: RUN “address_type_match_different_street_report.sql”

Will produce "address_type_match_different_street_report.lis" which is a listing of the records that match on address type with a previously loaded SCT Banner record (and both are active). SCT Banner only allows one active address per address type, so a decision needs to be made as to which record should remain active. Manual updating of SCT Banner or Legacy records is required for those that need to be made inactive. If Legacy records have been changed then the extracts must be re-run.

SPRTELE duplicate checking steps

STEP 1: Run “sprtele_duplicates.sql”

This program will check that the legacy telephone number has not been loaded previously for a specific telephone type. It will flag SPRTELE_CVT_STATUS with a “D” in the temporary conversion table SPRTELE_CVT. If the telephone number does not exist but other phone numbers of the same telephone type have been loaded preiously, the program will adjust the sequence number in CONVERT_SEQNO stored in the temporary conversion table SPRTELE_CVT.

STEP 2: RUN ”telephone_match_report.sql”

This program will produce a report named ”telephone_match_report.lis“ which will list the phone numbers that were found to be duplicate information based on telephone type.

Extracting Data from the Audit Tables

If you turn on the auditing feature in the control form (CUACTRL), perform the following steps to extract the rules changes from the CUBCNVT_AUDIT, CUBCNVT_AUDIT_VALUES, CURCNVT_AUDIT and the CURCNVT_AUDIT_VALUES tables in the form of update, insert or delete scripts.

1. Log in as sctcvt, or what ever the db login is that made the changes.
2. Install the 2 dbprocs (czkrule,sql and czkrul1.sql) under your convert user schema.
3. Run the dbproc to create the rule extract scripts by typing: execute czkrule.p_update_rules.
4. Run the create_get_rules.sql in the ctool/server/util directory. It will create a get_rules.sql.
5. Run get_rules.sql. This will create separate scripts for the CUBCNVT and CURCNVT table for each table owner that had changes to the rules in the CUACNVT form

Now you have the rules changes that have been made since the auditing feature was turned on

Validating Data for Crosswalked Columns

For many columns, a legacy value is being crosswalked using the function F_CVT_CURCVAL and the table CURCVAL. In many cases the legacy value is missing from the crosswalk table. The script is cvt_chkxwlk.sql, checks this after the data is loaded into the temp table. It takes two parameters, the SCT Banner table owner and the SCT Banner table name. It verifies that all the data in the CONVERT columns of the temp table exist in the CURCVAL table for the crosswalk. If a value doesn't, it inserts it into the crosswalk table, CURCVAL. It inserts the same value in the LEGACY_VALUE column and BANNER_VALUE column. You should then use the converter tool crosswalk form, CUACVAL to review the crosswalk entities for the table you are checking.

Example:

start cvt_chkxwlk SATURN SSBSECT

Validating Data for Columns with Foreign Key Constraints

Many columns have legacy values that are not crosswalked but still have a foriegn key constraint in a validation table. There is a script, cvt_chkfkeys.sql to look for these values and insert them into the validation table. It takes two parameters, the SCT Banner table owner and the SCT Banner table name.

Example:

start cvt_chkfkeys SATURN SSBSECT

Appendix H - SCT Converter Tool Functions

SCT Converter Tool Functions (alphabetically, by system) H-3

SCT Converter Tool Functions (alphabetically, by system)

Function Name	Description	Alumni	Financial Aid	Finance
F_CALC_INST_GPA	This function should be used as a wrap-up function on the last academic history table (shrtckg if only institutional records and shrtrce if inst. and transfer records are loaded).			
F_CVT_AFBCAMP_NAME	This function is used in an Alumni conversion to determine if the Campaign Name is longer than 30 characters.	X		
F_CVT_AMRSTAF_SOLICIT_IND	This function is to find the ATVSTFT_SOLICIT_IND for setting on the AMRSTAF record.	X		
F_CVT_AORCONT_SEQNO	This function is to find the next Organization Contact sequence number from the AORCONT table.	X		
F_CVT_APRACLD_LEAD_DESC	This function is to find the STVLEAD_DESC value for setting the Alumni Activity Leadership field APRACLD_LEAD_CODE.	X		
F_CVT_APRACLD_SEQNO	This function is to find the next sequence number for the Alumni Activity Leadership (APRACLD) record.	X		
F_CVT_APRADEG_SEQNO	This function is to find the next degree sequence number.	X		
F_CVT_APRCHLD_SEQNO	This function is to find the next child sequence number from the APRCHLD table.	X		
F_CVT_APRCHLD_XREF	This function is to ensure that we only populate the XREF field if the PIDM field is NOT null on the APRCHLD table.	X		
F_CVT_APRCSPS_XREF	This function is to ensure that we only populate the XREF field if the PIDM field is NOT null on the APRCSPS table.	X		
F_CVT_APREHIS_ATYP	This function is to find the SPRADDR record in order to set the ATYP_CODE on the APREHIS record.	X		
F_CVT_APREHIS_SEQNO	This function is to find the SPRADDR record in order to set the SEQNO on the APREHIS record.	X		
F_CVT_APRMAIL_ATYP	This function is to find the APBCONS_ATYP_CODE_PREF for setting the APRMAIL_ATYP_CODE field.	X		
F_CVT_APRXREF_PRI_IND	This function is to figure out what to set the APRXREF_CM_PRI_IND for the XREF record we are dealing with at the moment. **Note:** Make sure XREF has a SPOUSE record. We are only interested in doing this for spousal relationships. Parameter xref_code should be the converted banner value not the original plus value (ie; aprxref_rec.aprxref_xref_code).	X		
F_CVT_AUTO_ID	Used to verify that an ID exists for a record. If an ID is not passed to the function, then an ID will be generated.	X	X	X

Function Name	Description	Alumni	Financial Aid	Finance
F_CVT_AUTO_PIDM	********IMPORTANT******** *****MODIFY SPRIDEN_CONVERT.SQL***** *****ADD "ORDER BY" CLAUSE***** Purpose: Used to generate/verify that a pidm exists for a record. If a NULL convert_pidm is passed to the function, then a pidm will be generated from SOBSEQN since this will signify the most current record. If convert_pidm has a value (this means either a NAME change or an ID change is in the extract), then the pidm should not be generated from sobseqn. Rather, we will get the pidm from the current temporary table, spriden_cvt, where the convert_pidm is equal to the convert_id and the convert_change_ind is null. In order for this method to work, the current record must be processed first by the conversion program spriden_convert.sql (meaning: the cursor must have an 'ORDER BY' clause so that records with null change indicators are processed first. For example, ...order by convert_change_ind desc;...).	X	X	X
F_CVT_CHK_SPRADDR_SEQNO	This function finds out the sequence numbers for spraddr records of a given address type.	X	X	X
F_CVT_COLL_CODE	Purpose: This function can be used to fetch the college code into a student record from the SMRPRLE table when only the program code is known.			
F_CVT_DEGC_CODE	This function is used to fetch the degree code from the SMRPRLE table when only the program code is known.			
F_CVT_DEPT_CODE	Given the Major, college and degree, determine the department.			
F_CVT_DFT_ACTIVITY_DATE_CHK	This function will default the activity date to the system date if the legacy_value is null. Otherwise it will format and pass the legacy_value.	X	X	X
F_CVT_EITHER_CHECK	This function is to ensure that we do not populate both the PIDM field and the other legacy field on some of the ALUMNI tables.	X		
F_CVT_ENTITY_CODE	Given the Validation table name, create an Entity code for use in xwalking.	X	X	X
F_CVT_FIRST_TERM	This function is used to find the first term of attendance. The SGBSTDN table must be populated.			
F_CVT_GET_APRCATG_PIDM	This function is for use for alumni conversions. Once the legacy ID is put in aprcatg_cvt_pidm, we can find a constituents and organizations pidm by looking up the legacy id in aprcatg with a single record per pidm. The legacy ID is stored in the convert_pidm column. It works well so long as there is one record per pidm. If you convert all their alumni tables after converting one donor category per pidm, this should work fine. You can then convert the remaining donor categories into APRCATG.	X		
F_CVT_GET_NEXT_APPL_NO	To return the next application number in a series of Admissions applications.			

Function Name	Description	Alumni	Financial Aid	Finance
F_CVT_GET_PIDM	Purpose: Look at current id to get the spriden pidm if control file indicates using generated IDs look at the legacy ID (spriden_change_ind = 'I', SPRIDEN_NTYP_CODE = 'LGCY') to get the pidm. We are assume for conversion, legacy data that requires a pidm will use the legacy ID We are also assuming there is only one spriden_id with a change indicator of 'I' and name type code of LGCY If not using generated ID's, the most current spriden_id should be the same as the legacy ID so look for a change indicator of null	X	X	X
F_CVT_GET_SPRIDEN_CVT_PIDM	This function is for use for alumni conversions. Once the pidm corresponding to the legacy ID has been identified and logged stored in SPRIDEN_CVT.SPRIDEN_PIDM, we can then look for a pidm corresponding to a legacy ID by utilizing looking for a row with the specified legacy id in SPRIDEN_CVT.CONVERT_ID and return the pidm in SPRIDEN_CVT.SPRIDEN_PIDM.	X		
F_CVT_GET_SPRIDEN_LAST_NAME	Look at current or previous ID's to get the distinct last name. The parameter for this function will be tablename_rec.convert_pidm,where tablename is the name of the table you are working in. This will furnish the legacy ID to the function, which will then use the ID to match to the current spriden_id to return the last_name associated with the ID.	X	X	X

F_CVT_GET_VALUE	Feed in a column you would like to retrieve, along with a table you are retrieving from, and it will return the value from that table. Optionally, you can specify up to four where constraints (entering each where column with a where value), to ensure that only one row is returned. Example call, which retrieves the first name from SPRIDEN where the last name is 'Smith' and the ID is '12340'
	
	DECLARE
	my_value VARCHAR2(2000);
	BEGIN
	dbms_output.enable;
	my_value := f_cvt_get_value('spriden_first_name',
	'spriden','spriden_last_name','Smith','spriden_id','12340');
	DBMS_OUTPUT.PUT_LINE(my_value);
	END;
		X	X	X

F_CVT_GOREMAL_DISP_WEB	This function is to find if any of the Banner Web modules exists so we can set the GOREMAL_DISP_WEB_IND field.	X	X	X
F_CVT_GOREMAL_EMAL_CODE	This function is to find if person already has a pre-existing GOREMAL record for the EMAL_CODE before attempting to add current. It will also call a crosswalk function for emal_code.	X	X	X
F_CVT_MAJR_CODE	This function is used to fetch the major if the program is known. The SMRPRLE table must be populated.			

Function Name	Description	Alumni	Financial Aid	Finance
F_CVT_NAME_INITCAP	Translate the value passed in to have the initial character be upper case while the others are lower case. An exception is made for "Mc" or "O" where the third character will also be capitalized (McDonald or O'Keefe).	X	X	X
F_CVT_NULL_MEDI_NONNULL_DISA	Purpose: Given the Disa Code, and medi code is null... find matching Medi Code.			
F_CVT_NULL_SECD_ROLL_IND	Check if second major and return 'Y' so rolling second major information.			
F_CVT_PIDM_TO_ID	This function is for use for alumni conversions. Specify a pidm and this function returns the corresponding ID.	X		
F_CVT_PROGRAM	Given the Major, college and degree, determine the program.			
F_CVT_RETURN_NULL	This function will return a null value.	X	X	X
F_CVT_SET_SPRADDR_SEQNO	This function creates sequence numbers for spraddr records of the same address type.	X	X	X
F_CVT_SET_SPRTELE_ADDR_SEQNO	To set the sprtele_addr_seqno if a legacy value is not provided.	X	X	X
F_CVT_SET_SPRTELE_SEQNO	This function creates sprtele_seqno records where none are supplied.	X	X	X
F_CVT_TERM	To convert incoming legacy term codes to Banner Term codes. Used by SCT in testing. Can be modified at client site to fit client's legacy term code scheme. Alternative to building crosswalks in CUACVAL(CURCVAL).	X	X	X
F_GEN_RESET_SEQNO	Used to reset a sequence within the converter tool. Use SEQ1, SEQ2, and SEQ3. This function can be called during the breakpoint change.	X	X	X
F_GEN_SEQUENCE	Used to generate a sequence. For the converter tool, pass in SEQ1, SEQ2, or SEQ3.	X	X	X
F_GET_TCKN_SEQ_NO	To fetch the shrtckn_seq_no needed for academic history conversion. Required by shrtckl, shrtckg.			
F_VALIDATE_SINGLE	Powerful way to validate codes. Pass in the owner, table, and column of the validation table, along with the value you are verifying. The function will dynamically create a select statement based upon your parameters. The function returns 'FALSE' if no errors (it found the entry), and 'TRUE' if no data is found for the value you are passing in.	X	X	X
F_WRAP_SPRIDEN_EXISTS	use as wrap up functon for spriden when using generated ID's to see if the convert ID exists in production assumes we ran spriden_convert.SQL and pidm has a value that is either generated or from an existing spriden record if record in SPRIDEN_CVT is found with the same pidm in the BANNER spriden table set the status to 'X' so it will not get converted. USE WHEN GETTING A PIDM FOR SPRIDEN ONLY!! FOR ALL OTHER TABLES USE F_CVT_GET_PIDM Function that will return a NULL if ID does not exist rather than an error. This can be used in situations of running multiple stages of conversions where some of the ID's may already exist in the BANNER SPRIDEN table.	X	X	X
P_CVT_ERR	This procedure is used by the converter program to insert errors into the curcerr table.	X	X	X

Note: Many functions can be easily modified for use in other areas. Reviewing all functions is recommended to see if a modification to the function will help you achieve your goal. It is recommended that baseline functions are not modified, but copied and renamed.