Oracle Utilities Load Analysis

Load Data Management User's Guide

Release 1.11.1.2 for Windows

E18230-10

March 2019

ORACLE

Oracle Utilities Load Analysis Load Data Management User's Guide, Release 1.11.1.2 for Windows

E18230-10

Contents

Introduction

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 15

Chapter 16

Chapter 17

Introduction

Is This Guide for You?

The Oracle Utilities Load Data Management and Analysis System is a comprehensive software tool developed by Oracle Corporation to help utilities collect, manage, and analyze reliable load research data.

The *Oracle Utilities Load Analysis Load Data User's Guides* describe the concepts and procedures involved in working with the basic Oracle Utilities Load Analysis System. This volume, *Oracle Utilities Load Analysis Load Data Management User's Guide*, covers the Oracle Utilities Load Data Management Subsystem and is intended for anyone concerned with inputting, editing, managing, and/or reporting load data. The *Oracle Utilities Load Analysis Load Data Analysis User's Guide* covers the Oracle Utilities Load Data Analysis Subsystem and is written for utility statisticians and others concerned with applying various statistical analyses to the Oracle Utilities Load Analysis database.

Because this guide assumes no knowledge of Oracle Utilities Load Analysis or load research, it can be a useful self-teaching aid for anyone at any level.

How to Use This Guide

How you use this guide is up to you. You can either read the guide from beginning to end, or skip ahead to those chapters that pertain to your area of interest. (Each chapter begins with a brief overview of its contents, so you can quickly determine whether or not it is appropriate to your needs.) If you are a new Oracle Utilities Load Analysis user, however, it is recommended that you read all of the chapters in sequence.

Examples are provided throughout the guide to help you understand how the system works.

What You Should Know Before Getting Started

This manual is not intended to teach you the basics of working with your computer or operating system. If you need help with this, contact your facility's system manager.

Conventions Used in This Guide

The formats for creating input files are illustrated in boxes throughout the guide. Within these boxes, the following conventions are used:

- Key words that you will enter appear in the guide as a combination of upper- and lowercase letters. Typically, you need enter only the first three letters, which appear in uppercase.
- Parameters you will enter appear in italics.
- Braces { } are used to indicate a choice of parameters, from which you must choose one.
- Brackets [ ] are used to indicate optional parameters that you may or may not use.
- Vertical bars | separate mutually exclusive choices.
- Default parameter values (the values the Oracle Utilities Load Analysis system will use when you do not supply other values) are underlined.
- /* — the slash-asterisk combination identifies comments that do not affect processing. These comments are included in the sample files shown in this manual to help explain the examples. You can include similar comments in your own files by observing the following rules: Comments can be anywhere in a Control File or Environment File Statement. Each comment must begin with "/*". No commands will be recognized after a "/*".

  If you want to use the "/*" characters without them acting as comment designators, you must precede them with an ampersand: "&/*".

Throughout this manual, the following naming conventions are used to identify procedures and input/output files:

ELEMENT	CONVENTION	EXAMPLE
Input/Output Files	TGX___	TGX31A = Load Data Editor Control File TGX31B = Load Data Editor Environment File
Procedures	X___	X310 = Load Data Editor Procedure

Other Oracle Utilities Load Analysis Documentation

Some other references available from Oracle Corporation that you may find helpful include:

- *The Oracle Utilities Load Analysis Quick Reference Guide* — A concise summary of procedure names, input file commands and parameters, standard codes, and other essential information for the basic Oracle Utilities Load Analysis system and its extensions. It is a very useful tool to have at hand while you are working with Oracle Utilities Load Analysis.
- *The Oracle Utilities Load Analysis Load Data Analysis User's Guide* — The companion to this manual, *Oracle Utilities Load Analysis Load Data Analysis User's Guide* explains how to apply Oracle Utilities Load Analysis's various analysis programs to load data to calculate customer-, strata-, class-, and system-level statistics. Reporting and data management programs for the Load Analysis Subsystem are also covered.
- *Oracle Utilities Load Analysis Installation Guide/Oracle Utilities Load Analysis Configuration Guide* — Explain how to install, customize, and maintain Oracle Utilities Load Analysis as a standalone system, or as a client/server system in a LAN environment.
- *Oracle Utilities Load Analysis User's Guide* — Explains how to use the graphical user interface of Oracle Utilities Load Analysis to "submit jobs" — that is, how to select a desired program, create and specify the necessary input files, and view the results. This guide covers the mechanics of how to use Oracle Utilities Load Analysis. This guide should be used as a secondary companion piece to the *Oracle Utilities Load Analysis Load Data Management User's Guide* and the *Oracle Utilities Load Analysis Load Data Analysis User's Guide.*

In addition, there is a wide variety of optional program groupings, called "Bundles," that add additional capabilities to the base Oracle Utilities Load Analysis package. One or more of these programs may be in use at your facility. With the exception of a few programs documented in the Introductory Guides, most of the functions in each Bundle are delivered with their own manuals.

How to Get Help

Occasionally, as you work with Oracle Utilities Load Analysis you may encounter an error message or other problem that you cannot decipher on your own. As a Oracle Utilities customer, you can contact Oracle Support personnel at http://metalink.oracle.com.

My Oracle Support offers you secure, real-time access to Oracle experts on the complete Oracle Utilities Load Analysis system. It also provides ground breaking personalized & proactive support capabilities that help reduce unplanned down time and improve system stability. Leverage the Internet for immediate access to 24/7 support and get the critical and timely information you need for running your business.

Before contacting, please prepare the following information:

- Have available the outputs associated with the job (the job directory)
- Anything else that you think might aid in diagnosing the problem.

Chapter 1

Load Research and the Oracle Utilities Load Analysis System

This chapter is a useful introduction for anyone unfamiliar with load research and the Oracle Utilities Load Analysis System. It describes the basic concepts of load research, and explains how accurate load research can benefit a utility and its customers.

The chapter then gives a brief overview of the entire Oracle Utilities Load Analysis System — including the Load Data Management Subsystem, the Load Data Analysis Subsystem, and the functionality of each Bundle. Topics covered in this chapter include:

- **What Is Load Research?**
- **How Do Utilities Conduct Load Research?**
- **What Does Oracle Utilities Load Analysis Do?**

What Is Load Research?

"Load Research" is a way of determining how a utility's customers use electricity or gas — specifically, *how much* they use and *when*. Accurate load research is critically important to a utility, because it can help ensure cost-effective service, equitable rates, and profitability.

A utility's operations are determined by the amounts and times of customer usage — in other words, by the "load" placed on the utility's system. Customers require different amounts of energy at different times of the day and over the course of the year, and load patterns vary between different types of customers (e.g., commercial, residential, etc.). All of these customer demands combine to create a system load that fluctuates over time. Because energy cannot be stored during times of low consumption and released during times of high use — but instead must be produced as it is needed — a utility is required to build enough capacity to meet the highest load that could occur.

Load research enables a utility to get a precise picture of how its customers use energy. The utility collects "load data" for a representative sample of its customers. Load data measures customer demand at regular, short intervals (typically every 5, 15, 30, or 60 minutes). This data is analyzed and extrapolated to produce "load profiles." Load profiles (Figure 1-1) show how demand varies over different time periods and how each customer class contributes to the total system load. Accurate load research and load profiles can have many different applications, benefiting both the utility and its customers. For example:

- **Rate Design and Cost of Service Studies** — Load patterns cause cost patterns. Because the utility must build enough capacity to meet every customer's needs at the time of system peak, rates are often set according to how each customer class contributes to the peak load. Load research enables the utility to accurately determine how much it costs to provide service to each type of customer — ensuring that rates are commensurate with the level of service.

  Accurate load research helps protect the utility too; once costs are known, rates can be designed to collect enough revenue to cover all costs. In addition, public utility commissions are more likely to grant rate requests when accurate, reliable load data is available.

- **Demand Side Management** — Load research can help a utility and its customers save money too. Once load profiles are understood, customers can be encouraged to shift their usage from on-peak to off-peak. (In other words, the utility can flatten the load profile.) For the utility, this can mean a reduction in maximum capacity requirements and better use of existing resources. For the customer, this can mean lower energy bills.

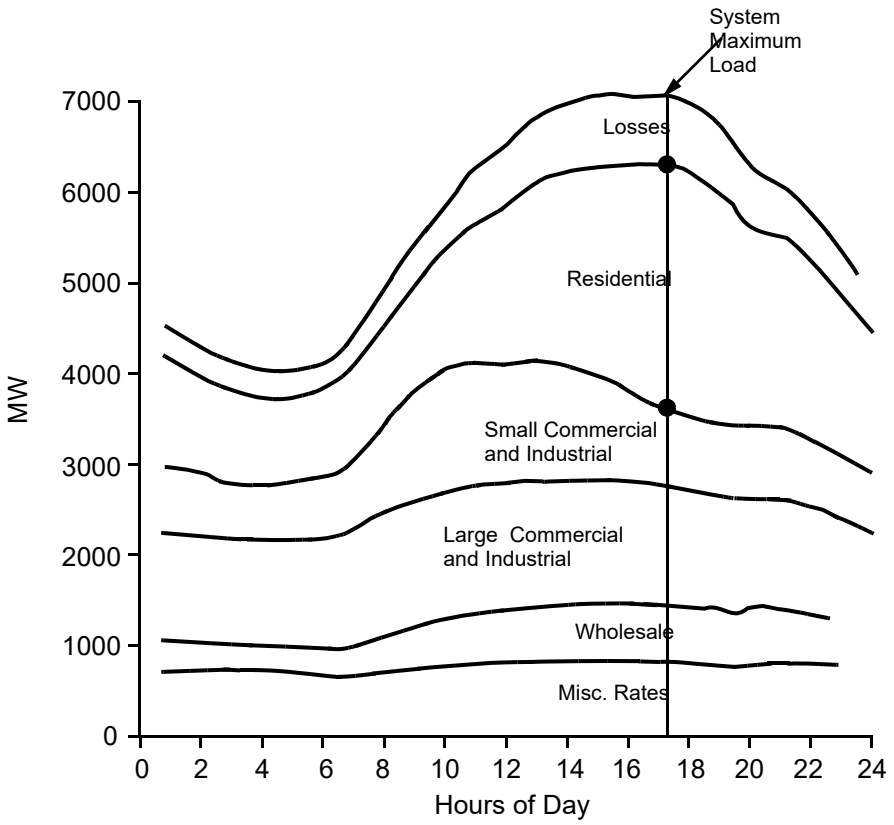


Figure 1-1 Load Profile

Load profiles are developed from load research data. They are very useful for forecasting system capacity requirements, encouraging energy savings through load management, setting equitable rates, and ensuring profitability.

- **Load and Energy Forecasting** — Load research helps utilities understand how different types of customers consume energy at different times (weekday v. weekend, work day v. holiday, winter v. summer, etc.). It helps utilities spot trends and prepare for the future.
- **Transmission and Distribution Planning** — T&D engineers can use load data to more effectively perform system planning, load balancing, and equipment selection.

How Do Utilities Conduct Load Research?

Each utility has its own way of conducting load research, but some general steps can be outlined.

First, the utility identifies its major customer classes — for example, residential, commercial, industrial, and agricultural. Then, a small sample group is identified for each customer class. This is necessary because it is prohibitively expensive to obtain load data for each and every customer. Of course all customers are metered to determine how much energy they consume during a given time period, such as kilowatt hours per month. Load research, on the other hand, requires measurement of demand at regular, short intervals — such as every 5, 10, 15, 30 or 60 minutes. Such measurement requires more effort and more sophisticated equipment — such as "smart meters" or "meter data management systems." Rather than installing expensive instruments at every customer site or performing analysis on the entire population, a a small but statistically-reliable subset of the total group is selected for monitoring. (This is referred to as "sampling.")

The monitoring devices continuously record the interval data at each selected customer site (Figure 1-2). Periodically, the data is collected and fed into translation or meter data management

software. These applications assemble the data into a “standard” format. However, because there are many proprietary recorders, meter data management, and translation systems available, there are many formats in use. Oracle Utilities Load Analysis accepts and recommends the use of the "Oracle Utilities Enhanced Input/Output Interval Data Format" (".LSE" files). This format ensures accurate and precise data is entered into the system.

The data is then transferred to Oracle Utilities Load Analysis, where it undergoes extensive validation and editing to make sure that it is complete and accurate. Once load data has been collected and validated for each sample group, it is analyzed and extrapolated to represent the entire population — providing an accurate picture of the amounts and time of energy consumption by each customer class.

What Does Oracle Utilities Load Analysis Do?

Oracle Utilities Load Analysis is a software application for the management and analysis of load research data.

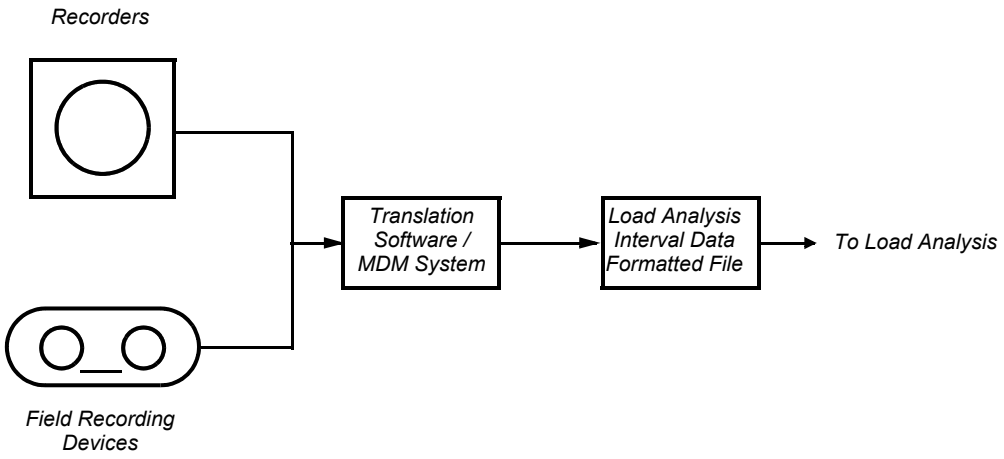


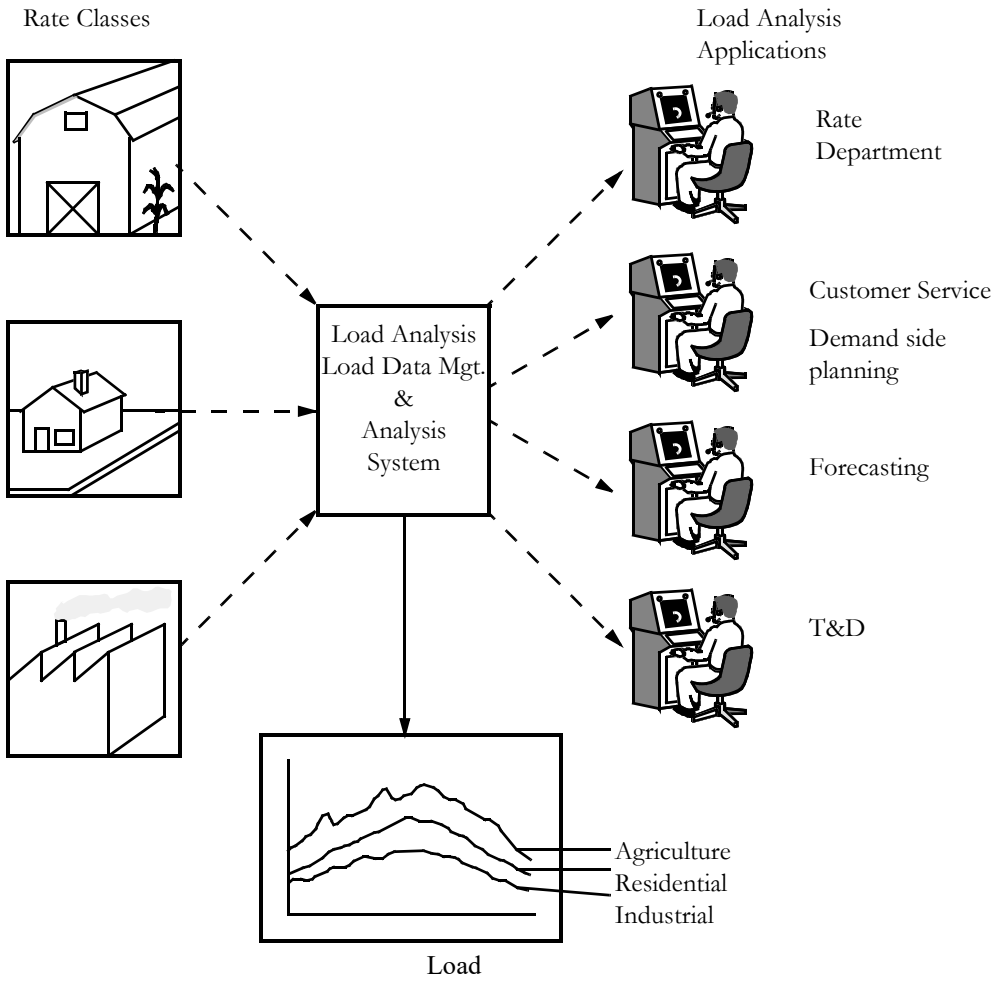
Figure 1-2 Load Data Collection

Recording devices collect interval data, which is assembled by translators.

Originally developed to help clients meet PURPA[1] requirements, Oracle Utilities Load Analysis has been enhanced and expanded over time to become a broad-based system for a variety of load research applications. Oracle Utilities Load Analysis is now in use at more than 100 utilities across North America, and is the most widely-used system of its kind.

1. PURPA — the Public Utility Regulatory Policies Act — was signed into law on November 8, 1978. It established a set of procedures and requirements for state utility commissions and electric and gas utilities and was designed to standardize rate setting and encourage conservation.

The basic Oracle Utilities Load Analysis package consists of two subsystems: *Load Data Management* and *Load Data Analysis*.



Load Data Management and Load Data Analysis; the Oracle Utilities Load Analysis Load Data Management and Analysis Systems make load data for customer classes available for a variety of applications and departments in the utility industry.

The functions of the **Load Data Management Subsystem** are:

- To accept load data from a variety of sources
- To ensure that the data is complete, consistent, and accurate
- To make it available for reporting and analysis (Figure 1-3).

The **Load Data Analysis Subsystem** applies complex formulas to the load data in order to generate meaningful statistics on customer load patterns, and it produces both standard and ad hoc reports. Both subsystems include programs for various "housekeeping" tasks such as data storage.

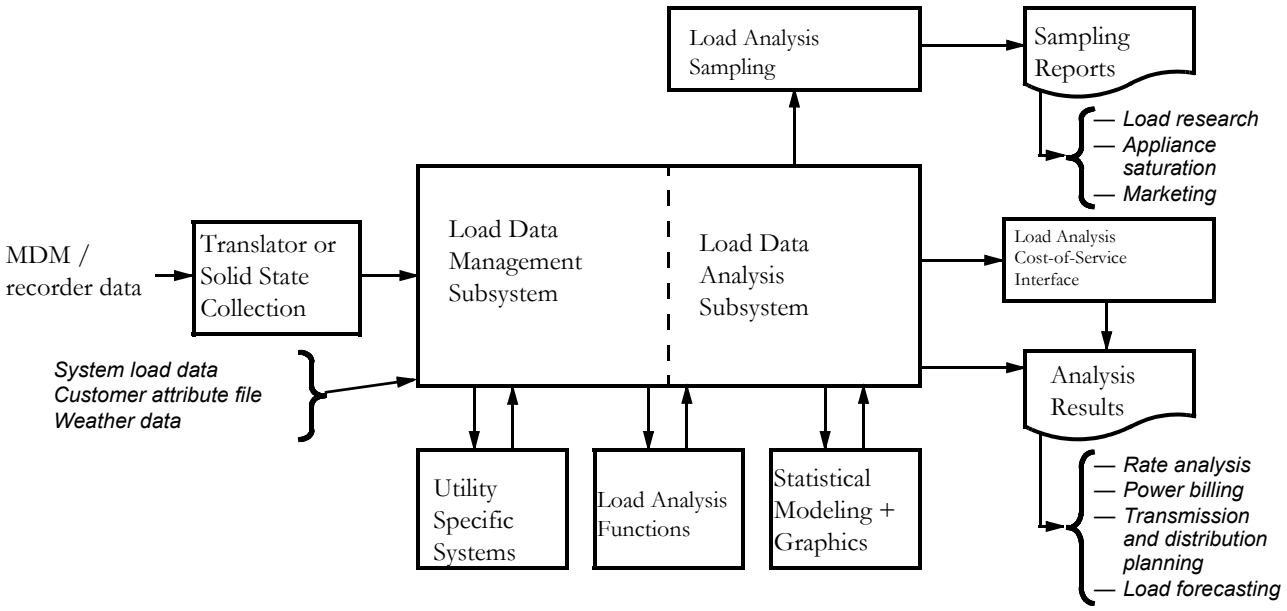


Figure 1-3 Overview of the Oracle Utilities Load Data Management and Analysis Subsystems

Oracle Utilities Load Analysis is a part of Oracle Corporation's Oracle Utilities Energy Information Platform. Oracle Utilities Energy Information Platform also includes Oracle Utilities Billing Component, Oracle Utilities Rate Management and its extension Oracle Utilities Rate Management - Revenue Forecasting, the Oracle Utilities Pricing and Settlement System, Oracle Utilities Quotations Management, and the Oracle Utilities Transaction Management, to name a few.

Analysis

Domains Analysis — enables analysts to produce estimates of total loads for subpopulations in an existing customer sample (for example, customers with air conditioning or grocery stores that are open 24 hours a day). This type of analysis is extremely useful for rate design, load management, forecasting, and marketing studies.

Individual Customer Analysis (ICA) — produces time-of-use, entire period and average day statistics and reports. Different time-of-use schedules can be applied to the same customer in a single run enabling what-if analysis. ICA produces statistics for interruption and load control periods, and can compute statistics for non-contiguous time periods in the same analysis.

Coincident Peak Analysis Program — makes it possible to produce estimates of rate class coincident peak demand and its corresponding error for up to 12 periods. It helps determine the responsibility of a rate class towards cost, providing a more accurate measure than approaches based on a single hour of demand. Coincident Peak Analysis uses both mean-per-unit and ratio estimation techniques.

Proxy Day Selection Program — selects a historical day (Proxy Day) whose load shape most accurately represents a supplied day's load shape. This selected day, in conjunction with the Cost of Service Interface, can be used to perform daily reconciliation for the estimation of aggregated suppliers' load profiles.

Reporting

Cost of Service Interface (COSI) — enables electric and gas utilities to balance sample class loads to system load and accurately allocate the cost of service among rate classes, sub-classes, and special industrial customers. Using COSI, analysts can build accurate demand profiles (incorporating transmission and distribution losses as well as unexplained sampling error) for each group and then apply cost allocation formulas to develop allocation factors. The resulting factors can then be fed into the utility's cost of service software for final calculations.

Totalizing Reporter — reports load profile data and summary statistics on multiple channels of data. It is particularly valuable for billing and analyzing large industrial customers, cogenerators, and wholesale customers with multiple points of delivery.

Late Cut Reporter — analyzes information that has been recently entered into Oracle Utilities Load Analysis and reports any cut series that are missing their last period data.

Validation Statistics Reporter — works as an addition to the Oracle Utilities Load Data Input and Validation modules. It examines and reports summaries of the results of internal validation tests performed on cuts of data entered into Oracle Utilities Load Analysis.

Sampling

Sampling Package — a tool for the design and selection of samples for any type of customer research, including load research and mail surveys. It supports both single- and multi-dimensional sampling, and can accommodate multiple variables (including both usage and demographics) within a single sample design. It is very useful for cost allocation, as well as marketing, forecasting, and demand planning.

Other Oracle Corporation Applications

Oracle Utilities Rate Management™ — a comprehensive tool to evaluate the revenue impact of alternative pricing rates for large volumes of customers, Oracle Utilities Rate Management is an extension of Oracle Utilities Rate Management - Revenue Forecasting. Oracle Utilities Rate Management accepts new rate components without reprogramming, maintains an audit trail of experimentation with proposed rates, and tracks migrating customers from one rate to another.

Oracle Utilities Billing Component™ — the Billing Engine is designed to meet the needs of large account billing and the complex contracts of commercial and industrial energy users. Oracle Utilities Billing Component uses a point-and-click Windows user-interface, performs automated bill calculation, integrates with all existing customer care systems and financial applications, and supports Enterprise Billing™.

Oracle Utilities Load Profiling and Settlement™ — an off-the-shelf software solution that addresses all components of the settlement process including: short term load forecasting; load data validating, estimating and editing; estimation of supplier contribution to system load; and daily and monthly financial settlement.

Chapter 2

Overview of the Oracle Utilities Load Data Management Subsystem

This chapter introduces the Oracle Utilities Load Data Management Subsystem programs and databases, and describes the steps you will follow when working with the subsystem. If you are a new Oracle Utilities Load Analysis user, you should read this chapter.

- **How Is the LDMS Structured?**
- **Working with LDMS — What Steps Do You Follow?**
- **LDMS Procedures**

The main purpose of the Oracle Utilities Load Data Management Subsystem (LDMS) is to build a reliable database of load data. The quality and reliability of the data is critically important to the success of your utility's load research program. For that reason, the subsystem has been designed to ensure a disciplined and consistent approach to data input and management. At the same time, a great deal of flexibility has been built into the subsystem so that it can conform to the specific needs of your corporation.

How Is the LDMS Structured?

The Load Data Management Subsystem consists of a group of programs and databases designed to enable you to input, validate, edit, store, and report load data. LDMS has two databases:

- **Current Load Database (CLDB)** — the raw, working database. It contains data that is in the process of being verified and edited, and the most recently collected load data. Typically, the CLDB contains three months to one year of load data.
- **Archive Load Database (ALDB)** — provides storage for data that has been verified as statistically reliable and ready for analysis. Users of the Oracle Utilities Load Analysis Subsystem extract data for analysis primarily from the ALDB.

Figure 2-1 illustrates the structure of LDMS. The following pages describe how you work with these programs.

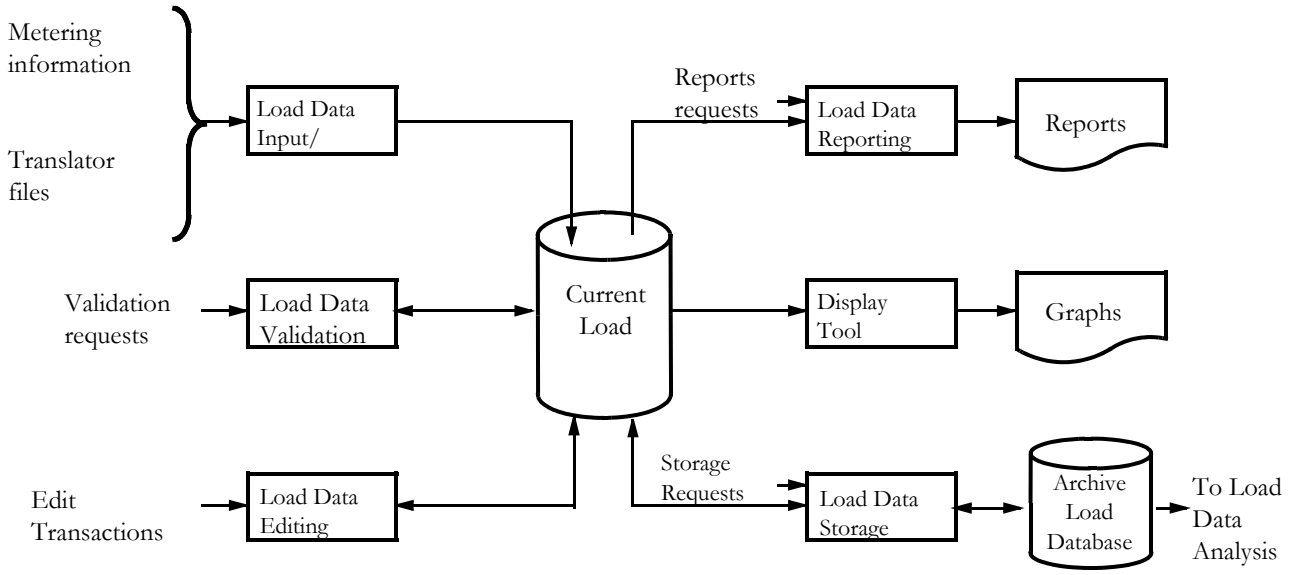


Figure 2-1 Structure of the Load Data Management Subsystem

Working with LDMS — What Steps Do You Follow?

The Oracle Utilities Load Analysis System is flexible enough to accommodate diverse and changing requirements yet disciplined to ensure data integrity. Thus, there are almost unlimited ways to set up and use the system, but once you have established a routine, Oracle Utilities Load Analysis handles much of the activity automatically.

For the sake of clarity, we will now look at the typical flow of data through LDMS (Figure 2-2).

Step 1: Data Input

"Load data" can incorporate many different measurements (kilowatt-hours, kilowatts, temperature, etc.), can originate from many different sources (solid state devices, digital pulse recorders, etc.), can be stored on different media, and in many different formats (makers of translation devices and software often use proprietary formats). The specifics will be unique to your utility and can be handled by Oracle Utilities Load Analysis.

The objective of this step is to get all of the data into a standard Oracle Utilities format and/or into the CLDB. Depending on the type and origin of the load data, you will use either the Manual Entry Program, the Load Data Input Program, a custom Input program handling many data formats, or the Oracle Utilities Energy Information Platform Adaptor.

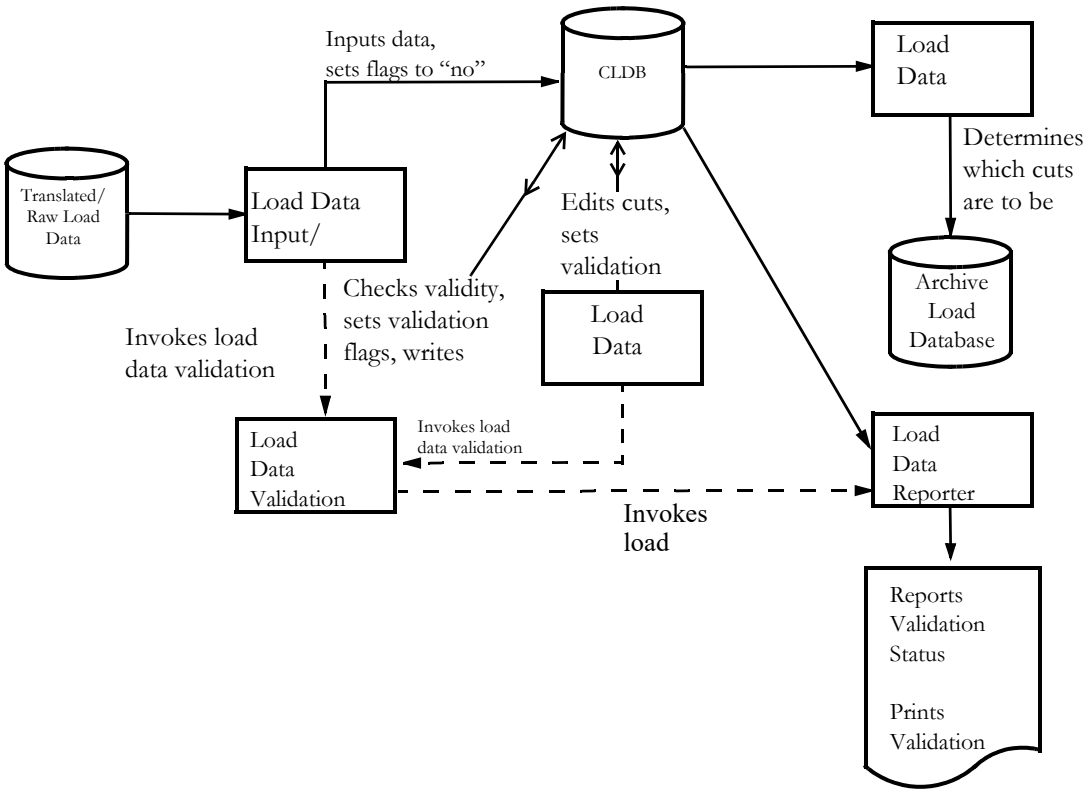


Figure 2-2 Flow of Data through the Load Data Management Subsystem

Step 2: Data Validation

The collection of load data is prone to many kinds of problems — power outages, missing data, translator problems, human error, etc. The objective of this step is to identify all "bad" or missing data.

Once you have input the load data into the CLDB, Oracle Utilities Load Analysis automatically runs a series of tests to check it for consistency and completeness. The Load Data Validation routines are used for these tests. There are two types of tests — "Internal Validation" and "External Validation." Internal Validation checks the quality of the load data itself; "External Validation" examines how the load data matches up with the preceding and following data records. The specific criteria for the tests are determined by your utility. Oracle Utilities Load Analysis flags all data that fail the tests as invalid and issues messages that you can use to identify, query, and correct the problem(s).

Step 3: Data Editing

The objective of this step is to correct any problems identified during Data Validation. The specific actions you take to correct the problems will of course depend upon the nature of the problems and the policies of your facility.

Oracle Utilities Load Analysis gives you three basic options for this procedure — you can use the Automatic Editor, which automatically edits data upon input, you can use the batch approach, which requires greater Oracle Utilities Load Analysis expertise but can in some cases save time, and you can use the web-based Interval Data Manager (IDM), which allows interactive load data editing.

Upon completion of this session, Oracle Utilities Load Analysis automatically reruns the validation tests. Again, all data records that fail are flagged as "invalid" and identified for re-editing. Thus, validation and editing are iterative processes that you run and possibly re-run until all data passes the tests.

It is important to note that for each record you edit, Oracle Utilities Load Analysis automatically creates three records:

- Inactive record — the original, unedited record
- Active record — the latest edited version
- Edit trail — a list of all modifications made to the data record.

In this way, the original data is always preserved and you have a record of all changes.

Step 4: Archiving

The next step in the process is to move the cuts from the CLDB to the ALDB, making them available for analysis by the Oracle Utilities Load Data Analysis Subsystem. To keep the CLDB to a manageable size, this procedure should be done on a regular basis. We recommend that you archive your data once a week or every month or set up a sequencer to do it automatically.

Typically, only internally and externally valid cuts are eligible for transfer to the ALDB. (You can, however, override this restriction in certain circumstances using the Load Data Editor.)

The Scan, Archive/Delete Procedure identifies the eligible cuts, moves them to the ALDB, and deletes them from the CLDB. The procedure moves all records associated with the cut — e.g., the active record, inactive record, and edit trails.

Step 5: Retrieval

Once cuts have been archived to the ALDB, you may find that you need to re-examine or re-edit them. Using the Retrieval Program, you can move them back into the CLDB, where you can view and/or modify them as necessary.

Step 6: Reporting

Upon completion of each program run, Oracle Utilities Load Analysis automatically generates a series of reports that list back the parameters of the run and any problems. *You should always check these reports carefully before going on to the next step*.

In addition, at any point in the process, you can generate a variety of useful reports on records in the CLDB. Oracle Utilities Load Analysis offers two primary reporting programs. *The CLDB Summary Reporter* enables you to produce a list of the entire contents of the CLDB as well as useful statistics on the database. With the *Load Data Reporter*, you can create comprehensive reports on individual customers — including peak summaries and detailed "data dumps." Using the Reporter in conjunction with the CLDB Key Generator Program, you can even generate reports on subsets of customers or records in the database, grouped according to user-defined criteria. Individual programs are also provided for summarizing and reporting cuts on the ALDB. Graphing this detail is also possible

LDMS Procedures

As we learned on the previous pages, the Oracle Utilities Load Data Management Subsystem consists of a series of programs and two databases. Oracle Corporation delivers these programs with a set of standard "execution procedures." These procedures are designed to run a series of programs in a logical, predefined sequence to accomplish specific tasks. For the purposes of this guide, we have identified the most important procedures. They are explained in detail in the remainder of this manual.

Chapter 3

The Oracle Utilities Load Data Format

In previous chapters we have often used the term "load data" to refer to the information that Oracle Utilities Load Analysis manages. In this chapter we will take a closer look at the kinds of data you will work with in the Load Data Management System (LDMS). We will also discuss the record types and flags the subsystem uses to manage the data as it goes through the input, edit, validation, and storage cycle.

This is a key chapter, and new Oracle Utilities Load Analysis users should be sure to read it. It defines many of the terms you will need to understand as you work with the LDMS, including:

- **Cuts, Cut Series, and Cut Keys**
- **Header and Interval Data**
- **Record Types and Flags**

To ensure data reliability and consistency, Oracle Utilities Load Analysis stores all load data in a standard format. This chapter explains each element of that format in detail.

Cuts, Cut Series, and Cut Keys

Oracle Utilities Load Analysis accepts and stores load data in units called "cuts" and "cut series" (Figure 3-1).

- **Cut** — A cut is a record of load data for a single customer and channel number, covering a discrete segment of time. It has a unique start-time and stop-time. Cuts are so-named because they represent a slice of time in the on-going measurement of a customer's demand. The length of time can vary from cut to cut, depending upon how the data was collected. The cut includes load data (also called "interval data" or "time-series data") and descriptive information. (Both are described on the following pages.)
- **Cut Series** — A cut series is a collection of cuts for a single customer and channel number, covering several consecutive time periods.

Typically, a cut is one month of data. Usually, the CLDB will contain cuts for the 3 or 4 most recent recording periods. It is possible to store more, but it is not advisable because too much data will slow down reporting and other procedures.

- **Cut Key** — A cut key is a unique identifier for a specific cut. You will often use cut keys when working with Oracle Utilities Load Analysis — to retrieve cuts or to create input files.

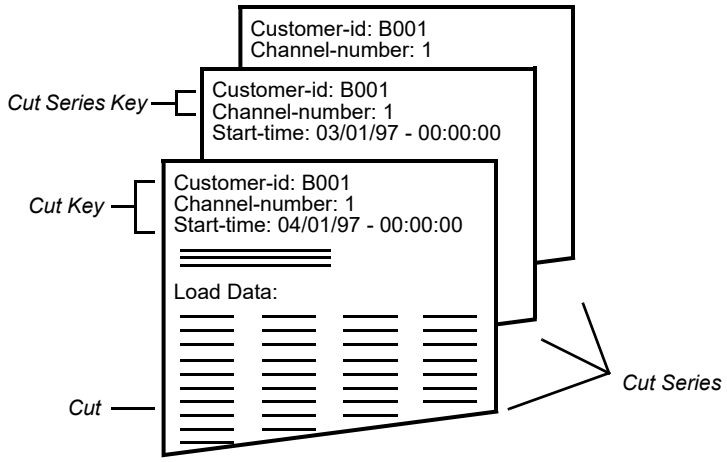


Figure 3-1 Cuts and Cut Series

A cut is a record for a single customer and channel number for a specified time period. A cut series is a collection of cuts for a single customer/channel number over several consecutive time periods.

A cut key always consists of the cut's "customer-id", "channel-number", and "start-time". (These terms are described on the following pages.) For example, the cut key for customer "B001", channel "1", with a start time of July 1, 1998 at 00:00:00 would be:

```
B001, 1, 07/01/98-00:00:00
```

When entering the cut key, you *must* input customer-id, channel-number, and start-time in that order. You may insert a comma and/or blank space between the elements or format them as above.

- **Cut Series Key** — The identifier for a cut series is called the cut series key and consists of the "customer-id" and "channel-number". For example, the cut series key

```
B001, 1
```

would identify *all* cuts in the CLDB with the customer-id "B001" and channel 1, regardless of their start-times.

Header and Interval Data

A Oracle Utilities Load Analysis cut consists of two parts:

- **Header** — a series of fixed-length fields that contain descriptive information about the cut.
- **Interval Data** (also referred to as "time-series data" or "load data")— a series of values that represent customer energy use — or factors that influence that use, such as air temperature — measured at regular, short intervals of time. While these values represent kW, kWh, degrees, or other meaningful measurements, they are typically stored as "energy," which can be translated into kilowatts or other meaningful measurements. The interval data portion can be any length.

Figure 3-2 shows a sample Oracle Utilities Load Analysis cut as you would see it in report format. Each element of the cut is described in detail on the following pages. (***Note***: The record format for cuts is described in *Appendix B: Record Formats Used by Oracle Utilities Load Analysis Input Programs*.)

- **Customer-id** is a unique 1- to 64-character identifier assigned to each customer or recording device by rate analysts or others at your utility.

  Typically, the ID is assigned to the recording device when it is installed, and Oracle Utilities Load Analysis automatically preserves the ID with the cut. However, Oracle Utilities Load Analysis gives you the option of modifying the ID during the data input process. This modification can be powerful as you can name your cuts so that future queries are possible.

- **Channel-number** is an integer (0 - 32767) used to identify the port, register, or channel of the device used to record load data for a customer. As with customer-ids, the channel-numbers are usually assigned when the device is installed, but it is also possible to change the numbers during data input.

- **Start- and Stop-Times** define the beginning and end of the cut's recording period. The standard format for expressing start- and stop-times is:

```
mm/dd/yy-hh:mm:ss
```

where:

```
mm = month, 01 ≤ mm ≤ 12

dd = day, 01 ≤ dd ≤ the number of days in the respective
month

yy = year, (19)67 ≤ yy ≤ (20)55

hh = hour, 00 ≤ hh ≤ 23

mm = minute, 00 ≤ mm ≤ 59

ss = second, 00 ≤ ss ≤ 59
```

In some cases, the format mmddyyhhmmss may also be used.

> **Important Note**: Standard start-times usually have seconds set to :00. Example: A cut that starts at the beginning of a day has a start-time of 00:00:00. If this cut ends at the end of the day (midnight), then the stop-time should be 23:59:59.

For example, the start-time for a recording period that began at midnight on July 1, 1998 would be:

```
07/01/98-00:00:00
```

- **Descriptor** is an 80-character, user-defined field. Typically, it is used to store customer name, address, and account number. Oracle Corporation can define the field to allow queries on specific information.

- **Time Span** is the time period covered by the cut and the number of intervals in the cut. These fields are automatically computed by Oracle Utilities Load Analysis.
- **Seconds-per-Interval** is the length of each recording interval. The most common interval lengths are 300, 900, 1800, and 3600 seconds.

  Through a technique known as "aggregation," data can be combined and reported at a higher frequency. For example, 1800-second data can be aggregated to 3600-second data. The reverse is possible.
- **Meter Data, Multiplier, and Offset** — many utilities record metered energy along with the interval data. This makes it possible to check the accuracy of the interval data in the cut, by comparing the total interval data with the total metered energy. Other utilities — especially those that rely on solid state devices — may not record metered energy. These are optional fields.

  Oracle Utilities Load Analysis stores the meter start-reading and stop-reading as integer values, associated with two constants called the *meter multiplier* and the *meter offset*. The meter multiplier and offset are set when the meter is installed, and will not change unless the device is modified. Meter multipliers must be positive; meter offsets are frequently zero, but can be positive or negative.

  The term "**meter energy**" is used to refer to the real measurement value associated with a net-meter-reading (the difference between the meter-stop-reading and the meter-start-reading). You can calculate meter energy by using the following formula:

$$\text{Meter Energy} = \left[\begin{pmatrix} \text{Meter Stop Reading} \end{pmatrix} - \begin{pmatrix} \text{Meter Start Reading} \end{pmatrix} \times \text{Meter Multiplier} \right] + \text{Meter Offset}$$

- **Interval Data** (also called "time-series data" or "load data") is a series of values that represent load data or other measurements taken at specific intervals (see lower section of Figure 3-4).

  The interval data can be reported as demand or energy values.

  > **Note:** Except for the intervals containing the start- and stop-times, each interval is referenced by a date/time that *ends* one second before a *whole interval boundary*; i.e., a time evenly divisible by the interval length expressed in minutes per interval. For example, :15:00, :30:00, :45:00, and :00:00 are whole interval boundaries for 900-second data. All times not one second before a whole interval boundary will be rounded up to one second before the next highest boundary; e.g., for 300-second data, the time 09:51:00 will become 09:54:59.
- **UOM** stands for "Unit of Measure" and identifies the type of interval data stored in the cut. UOM is often represented by a 3-digit code. For example, "01" means that load data is stored as kilowatt-hours. A complete list of the codes used in the Oracle Utilities Load Analysis system is provided in *Appendix A: Oracle Utilities Unit of Measure Codes*.
- **Pulse Multiplier and Pulse Offset** are values used to convert pulse counts in an Oracle Utilities Load Analysis .INP file into real measurements. The pulse multiplier is a scale factor, and the pulse offset is a displacement. The values can be different for each customer-id/channel-number. These values are stored for reference only, provided a cut was created from an .INP file containing them. Zeros in both of these fields indicates that they were not used in creating the cut, which may instead have been created using the .LSE input format.

  > **Note:** Pulses are not stored in the database. All pulses from an incoming .INP file are converted to recorded values. However, the pulse multiplier is retained for the sole case of extracting the original pulse values back into a .INP file. If a cut with a populated pulse multiplier is ever edited or changed, the original pulse multiplier is lost.
  >
  > Oracle does not recommend the use of pulse data as input.

- **Interval Energy** is the total amount of energy used by the customer during the time period spanned by the cut.
- **Validation Messages** are attached to the cut by Oracle Utilities Load Analysis during the validation process. They are diagnostic messages that alert you to "bad" data; e.g., data that failed one or more of Oracle Utilities Load Analysis's validation tests.
- **Edit Trail** lists all edits made to a cut. The Edit Trail is shown as a field in the report, but each Edit Trail is actually saved as a separate record in the CLDB, identified by the same cut key as the original cut.
- **Status Codes** indicate the quality of the interval data. There is one status code for each data value. The collection of load data is prone to different problems — for example, human error, power outages, timing channel malfunctions, etc. It is important during load analysis and other activities that the reliability of the data is known.

  The recording devices themselves assign status codes to the data. However, each device manufacturer uses a different set of conventions. Oracle Utilities Load Analysis may automatically translate these codes to a consistent standard during the data input process. Figure 3-2 shows a list of the Oracle Utilities Load Analysis status codes, arranged in order of increasing uncertainty. ***Note***: Alpha codes generally indicate that the data is acceptable; numeric codes indicate that the data is suspect and requires correction.

Status Code	Classification of Interval Data
' '	Normal
'A'	Normal, alternate-recorded (e.g., hand-entered)
'J'	Data inserted by Load Analysis to correct outage
'L'	Default for data modified by Load Data Editor
'N'	Interruptible or curtailable load
'P'	Inserted outage
'Q'	Corrected outage
'X'	Cuts resulting from merging invalid data or from unrecognized status codes
'Y'	Reserved
---	---
'	
1'	Uncorrected outage (also called loss of potential)
'2'	Non-normal (usually timing-pulse defects)
'5'	Aggregated interval used in rolling format with partially missing or unavailable data
'7'	Aggregated or transformed interval with partially missing data
'9'	Missing

Figure 3-2 Oracle Utilities Load Analysis Status Codes

The manufacturer's codes are translated to Oracle Utilities Load Analysis status codes during Load Data Input. This translation is designed to be as consistent as possible between various manufacturers. Data that is altered during Load Data Editing is assigned a different code, depending upon what was done to it. For example, normally-recorded, unedited data is assigned a blank status code (' '), which is the best quality code on the list. Inserted data is assigned a 'poorer' status code ('J') but ranks above patched data ('L'). The boundary between two successive recording periods in a series is a potential trouble spot. When using a rolling format, data aggregated from partially missing data or unavailable data is assigned a status code of ('5'). Data aggregated or transformed from partially missing data is assigned a status code of ('7'). Missing data carries the worst code of all ('9') and is therefore distinguishable

from actual zero usage. Only numeric status codes ('1' - '9') are flagged during the validation process.

Note: Your utility may have specialized status codes in addition to the standard Oracle Utilities set. The following box is provided for you to list the unique status codes used at your facility:

Status Code	Classification of Interval Data

About Daylight Saving Time and Interval Data

Oracle Utilities Load Analysis handles *daylight saving time* by storing data on a local time basis. The load data for the spring day is assumed to span 23 whole hours. The spring DST change is the last Sunday of April before 1987. 1987 and after, it is the first Sunday of April. The load data for October is assumed to span 25 hours. Follow the 2007 and beyond appropriate rules. Adjustments are automatically made to ensure the proper evaluation of data that spans daylight saving time boundaries.

Record Types and Flags

In addition to all of the information described on the preceding pages, a Oracle Utilities Load Analysis cut also includes a field to indicate record type and five flags to indicate its status. Oracle Utilities Load Analysis uses these fields to manage the progress of the cut through the input-validation-edit-store cycle. They are automatically set by Oracle Utilities Load Analysis — you do not change them except under special circumstances.

Record Type

The LDMS stores three types of records in the CLDB for an edited cut. All three records are identified by the same cut key, but they are distinguished by the "Record Type Field."

- **Active** — The most recent version of a cut; indicated by a blank in the record type field, or the word "ACTIVE" on reports.
- **Inactive** — The original version of an edited cut; indicated by an 'A' in the record type field, or the word "INACTIVE" on reports. Inactive records exist only for edited cuts.
- **Edit Trail** — A list of any and all edits made to a cut; indicated by a 'T' in the record type field.

When you first enter cuts into the CLDB, Oracle Utilities Load Analysis automatically stores them as active records. For each cut you edit, Oracle Utilities Load Analysis automatically creates the three records listed above (Figure 3-3). This maintains a record of the original cut and a list of all changes made to it. ***Note***: If you edit a cut more than once, Oracle Utilities Load Analysis still stores only the original and the most recent version. The intermediate versions of the cut are not stored. However, an Edit Trail is kept for every modification.

There is a maximum of 200 edit trails per cut.

> Only active records are available for validation, but both active and inactive records are available for reporting and editing. Again, they are identified by the same cut key, but they have different values in the Record Type field.
>
> Only CLDBs and ALDBs support this functionality.

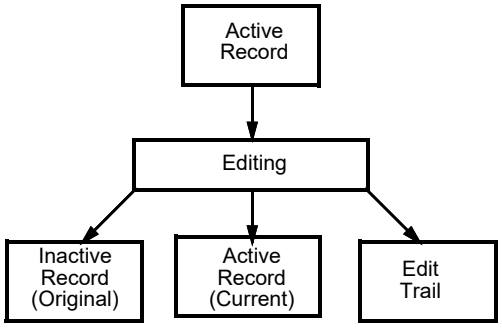


Figure 3-3 Record Types in the CLDB

The LDMS automatically creates three records for each edited cut. In this way, the original cut and a list of all edits are always available.

Flags

The flags indicate the validity of the load data in the cut and serve to keep "bad" or incomplete data from contaminating the ALDB and ELDB (Figure 3-4). The flags are:

- **Internally valid** — This flag indicates whether or not the cut has passed a series of "internal validation" tests. These tests examine the status codes, start- and stop-times, interval values, and meter readings associated with the cut. Oracle Utilities Load Analysis automatically sets this flag during the Validation process.
- **Externally valid —** This flag indicates whether or not the cut has passed a series of "external validation" tests. These tests determine whether or not there are gaps or overlaps between the interval data in the cut and the cuts preceding and following it in the cut series. Oracle Utilities Load Analysis automatically sets this flag during the Validation process.

  > **Note:** Oracle Utilities Load Analysis will always flag the most recent cut in a series as externally invalid, because there is no following cut to compare it with.

- **Merge** — This flag indicates whether or not the cut is OK to merge — i.e., OK to analyze. You can set the Merge flag to "Yes" during Editing, but the preferred procedure is to allow Oracle Utilities Load Analysis to automatically set it during the Archive process.
- **Archive** — This flag indicates whether or not the cut is OK to archive — i.e., OK to put in the ALDB. A cut can be OK to archive even if it is not OK to merge. You can set the Archive flag to "Yes" during Editing, but the preferred procedure is to allow Oracle Utilities Load Analysis to automatically set it during the Archive process.
- **Edited** — This flag indicates whether or not the cut has been edited. A "Yes" in this field tells Oracle Utilities Load Analysis to look for the associated edit trail(s).

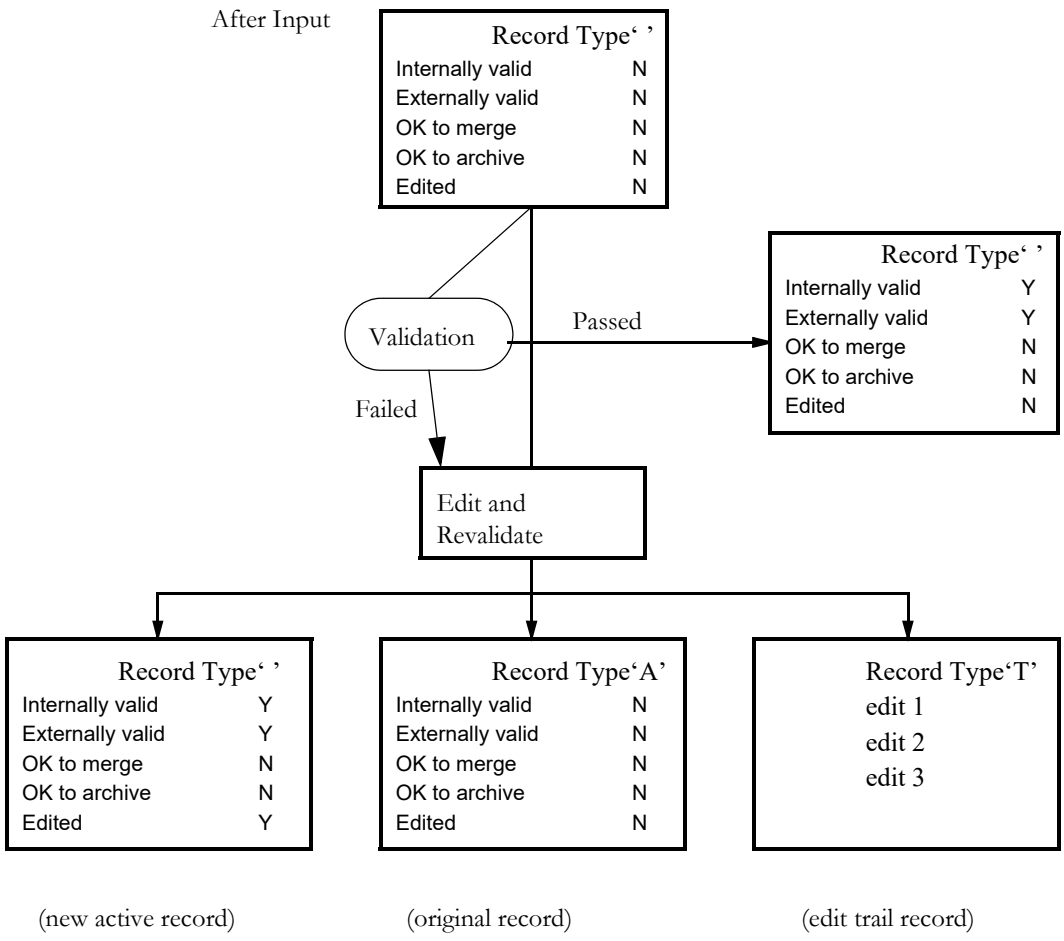


Figure 3-4 Record Types with Flags

For each cut, Oracle Utilities Load Analysis automatically controls the record type, edit, validation, and merge and archive flags. The above illustrates how Oracle Utilities Load Analysis creates new records and modifies flags after the data input and validation steps.

Chapter 4

Oracle Utilities Load Analysis Mechanics

This chapter is intended primarily for anyone unfamiliar with the basic concepts of input and output files. However, the chapter also defines some important terms unique to Oracle Utilities Load Analysis, such as "Environment files" and "Control files." Topics covered in this chapter include:

- **What Input Files Are Used in Oracle Utilities Load Analysis?**
- **What Output Files Does Oracle Utilities Load Analysis Produce?**
- **Naming Conventions in Oracle Utilities Load Analysis**
- **Using the Preprocess Key Generator in a Control File**
- **Querying the Database with Embedded SQL**

Before you begin working with Oracle Utilities Load Analysis, you might want to review some of the basics of working with the programs on your computer system.

You typically accomplish tasks by submitting custom command files or Oracle Utilities Load Analysis "jobs" that run programs. Each program requires input files and produces output files. Output files may be used as the final result (reports, for example), or may become input for other programs.

Another way of looking at this is that you can accomplish the tasks by following three main steps:

1. Creating Input Files
2. Running the Program
3. Checking Output Files.

What Input Files Are Used in Oracle Utilities Load Analysis?

Most Oracle Utilities Load Analysis programs use two types of input files: *Control files* and *Environment files*.

- **The Control File** (.CTL) typically contains the data to be processed. Customer lists and lists of keys for load data records are common kinds of Control files.
- **The Environment File** (.ENV) contains your processing instructions to the program. Typically, you select among a set of optional parameters to define the file.

Some programs also use one or more of the following files: *Time of Use File* (.TOU)*, Holiday File* (.HOL)*,* and/or auxiliary *database (AXDB)*. The first two are typically set up at system installation. The AXDB is explained in *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)*.

Output — the result of running a program — can be new or modified data entered into a database, Interface files, or reports. Aside from special reports, several reports are automatically produced by Oracle Utilities Load Analysis to help you evaluate the success of the program run. (These reports are typically bundled into a single output file called REPORT.HTML.)

- **The environment report** describes the processing mode in effect during the program run. Use this report to verify that you entered the correct commands and parameters in the Environment File.
- **The execution log** typically lists data that was altered or created during the program run; it often includes diagnostic messages that alert you to any problems with the input data or the run.
- **The summary report** usually provides counts of records entered or modified in the database as a result of the run.

What Output Files Does Oracle Utilities Load Analysis Produce?

Output — the result of running a program — can be new or modified data entered into a database, or reports. Aside from special reports, several reports are automatically produced by Oracle Utilities Load Analysis to help you evaluate the success of the program run.

Naming Conventions in Oracle Utilities Load Analysis

Elements of a Oracle Utilities Load Analysis program are identified by a consistent set of naming conventions. Being able to recognize programs and input files by these naming convention will be a big help to you in working with the system. Here are the conventions used in the Load Data Management Subsystem:

Element	Convention	Example	Typical Location
Programs	TGX_ _ _.JAR	TGX310 = Load Data Editor	Server
Control files	TGX_ _A.CTL	TGX31A = Load Data Editor Control File	Client, Common Data
Environment files	TGX_ _B.ENV	TGX31B = Load Data Editor Environment File	Client, Common Data
Output files	TGX_ _1.DAT	TGX311 = Load Data Editor Output File	Job Directory

Output files can be input files to other programs. In that case, the identifiers are assigned according to what program they are going into. For example, the Validation Control File from Editing is identified by "TGX21A.CTL".

The "X" that appears consistently above refers to elements of the Load Data Management Subsystem (LDMS). The convention for programs, procedures, and files in the Load Data Analysis System (LAS) uses a "Y" in the same position.

> These are the conventions Oracle Corporation assigns. You are not required to use these conventions when executing the programs — you can assign identifiers of your own. However, a logical set of identifiers is useful.

Using the Preprocess Key Generator in a Control File

Traditionally, using a key generator query to run a report required submitting two separate jobs. For example, to run the X440 program using a list of generated keys you had to:

1. Submit an X810 Key Generator job.
2. Submit an X440 Summary Reporter job using the results returned from the X810 job.

This example requires the following steps:

1. Create the X810 CTL file.
2. Run the X810 Key Generator.
3. Wait for the job to complete.
4. Copy the contents of the generated output list and paste it into the X440 CTL file.
5. Run the X440 Summary Reporter.

Now you can use the Preprocess Key Generator to consolidate these steps into one job by embedding key generator logic into a control file. Doing so allows you to dynamically create a list of full or partial keys to be used by the program.

For example, you can include the following key generator commands in your X440 control file to run the Summary Reporter on all cuts with a CUSTID that begins with "N":

```
L1: SUBSTR(CUSTID,1,1) = 'N' T(PRINT 1) F(NEXT)

FORMATS:
1: TRIM(CUSTID) ', ' TRIM(CHANNEL)
```

Note:

For programs that can run against multiple databases, the preprocessor will only run against the primary database selected (CLDB if selected, ALDB if no CLDB is available).

If an error occurs during the key generator query process, the job will be aborted with return code 99. Key generator outputs (such as sysprints and reports) will remain in the job output folder. You can use these files to troubleshoot the error that occurred.

For more information about the key generator program see [cross reference to LDM Chapter 15]

Preprocessing Options

You can use the following optional features with the preprocess key generator:

Keys List (Optional)

The keys list command provides the same functionality as the optional keys list file used by the key generator program. This command allows you to filter the results of the pre-processing query to a select subset of keys.

To supply an optional key generator keys list, enter the following statement at the end of your pre-process control file:

```
KEYSLIST:
```

Following the KEYSLIST statement, supply your list of keys (full or partial) in the subsequent lines using the following syntax:

```
KEYSLIST:
custid channel [start-time]
```

Example CTL file with KEYSLIST statement:

```
L1: SUBSTR(CUSTID,1,1) = 'N' T(PRINT 1) F(NEXT)

FORMATS:
1: TRIM(CUSTID) ', ' TRIM(CHANNEL)

KEYSLIST:
N1723, 1
N1725, 1
```

Environment File (Optional)

The Preprocess Key Generator uses the tgx81b.env file in the COMMON/DATA directory as a default environment file. If this file is not found, the key generator will create one with the following default value:

PRINT ECONOMY

You can override this setting to provide a custom environment file by using the following statement at the end of your pre-processing control file.

```
ENV:
```

Following the ENV statement, enter your environment commands. The following is an example:

```
/* FIND CUTS WITH MORE THAN 1 EDIT (EACH EDIT PRODUCES 2 TRAILS)
L1: TRAILCT > 2 T(PRINT 1) F(NEXT)

FORMATS:
1: TRIM(CUSTID) ', ' TRIM(CHANNEL) ', ' TRIM(START)

KEYSLIST:
N1723, 1
N1725, 1

ENV:
TRAILS
PRINT ECO
```

Supported Programs

The following Load Analysis programs support the Preprocess Key Generator:

- X210 Cut Series Validation
- X310 Load Data Editor (CLDB)
- X410 Load Data Reporter
- X430 Totalizing Reporter
- X440 Summary Reporter
- X470 Late Cut Report
- X480 Validation Statistics Reporter
- X490 Cut Series gap Reporter
- X530 Cut Series Overlap reporter
- X720 Direct Output (CLDB)
- Y230 Billed Energy Program

- Y240 Load Data Extraction
- Y310 Standard Load Data Analysis
- Y320 Aggregate Load Analysis
- Y330 Ratio Analysis
- Y350 Domain Analysis MPU
- Y360 Domains Analysis Ratio
- Y370 Individual Customer Analysis
- Y380 100% Sample Analysis
- Y410 Time Series Reporter
- Y620 Load Data Transformation
- Y630 Load Data Editor (ELDB)
- Y720 Direct Output (ELDB)
- Q91C Copy Cuts
- Q91D Delete Cuts
- Q91M Move Cuts

Querying the Database with Embedded SQL

The key generator has traditionally been used to query the database to create a list of cuts or cut series for input into programs. The key generator works well for complicated query logic, but it can be inefficient when used to search for cuts in a large database. For that situation, a better way to query the database is with embedded SQL Statements.

SQL statements can be embedded in an input control or key list file for programs that take a list of cuts or cut series as input. The SQL is used to query the database and return a list of cut or cut series before the a program is run. The control file or key list file that contains embedded SQL will have the SQL statements replaced with the results of the query before the main program is processed.

Syntax for Embedded SQL Statements

All embedded SQL statements are identified by the keyword @SELECT and use the following syntax:

```
@SELECT selectFields WHERE whereConditions

[KEYSLIST:]
```

Where:

selectFields are a comma-separated list of cut values to return. See **Selection Field Names** for a list of available field names.

whereConditions are a set of conditions from which to use as criteria for selection. The where conditions have the following format:

```
FIELD comparison-operator VALUE
```

Where:

FIELD specifies the field name which will be the target of the selection criteria. See **Selection Field Names** for a list of available field names.

comparison-operator specifies the logic to use for the comparison. See **Comparison Operators** for a list of available operators.

VALUE specifies a value to compare the field to. Values can be of type string, numeric, or Boolean.

Values should be in the following formats:

- String data types must be enclosed in single quotes ('example').
- Boolean values must be 'YES', 'Y', 'TRUE', 'NO', 'N', or 'FALSE'.
- Numeric values (NUM values) must be numeric with no plus (+) signs. They can either be in single quotes or not.
- Dates must to follow the standard Oracle Utilities Load Analysis convention (MM/DD/YY-HH:MM:SS or MM/DD/YY). They can either be in single quotes or not.
- Lists need to be enclosed in parentheses and values separated by commas. For example:

  ```
  CUSTID IN ('N1723', 'N1724')
  ```

The following is an example of an embedded SQL query statement:

```
@SELECT CUSTID, CHANNEL WHERE CHANNEL = 1
```

You can set multiple where conditions by joining them using AND and OR statements. Sets of conditions can be grouped together by enclosing them in parenthesis "()".

For example, to select a list of cuts in which the CUSTID is equal to "N1723" and CHANNEL is equal to 1, or in which the CUSTID is equal to "N1725" and CHANNEL is equal to 1, you would use the following statement:

```
@SELECT FKEY WHERE (CUSTID= 'N1723' AND CHANNEL=1) OR (CUSTID'N1725'
AND CHANNEL=1
```

KEYSLIST is a list of full or partial keys used to filter the results of the query. The KEYSLIST command must be enter following the embedded query statement as follows:

```
@SELECT selectFields WHERE whereConditions

KEYSLIST:
custid channel [start-time]
```

See **Keys List (Optional)** on page 4-4 for more information about the KEYSLIST command.

Escape Characters

Single quotes, underscores, and percent symbols are special text characters. Single quotes are used to designate the beginning and end of a text field. Underscore and percent symbols are used as wildcard characters. To represent text with these characters literally, they will need to be escaped:

To Use Single Quote: Use double single quotes to represent text with apostrophe or single quote literal.

Example:

```
@SELECT PKEY WHERE DESCRIPTOR LIKE '%MARK''S PIZZERIA%'
```

To Use Underscore and Percent Sign: Use the backslash character ("\") to escape underscore and percent symbols. This applies only when using the LIKE operator, where "_" and "%" are wildcard characters.

Example:

```
@SELECT PKEY WHERE CUSTID LIKE 'MONDAY\_CUT'
```

Selection Field Names

The following table lists the valid field names that can be used in embedded SQL queries:

Field	Type	Description	Clause
FKEY	Text	Returns a list of cuts, or full keys (CUSTID, CHANNEL START).	SELECT
PKEY	Text	Returns a list of cut series, or partial keys (CUSTID, CHANNEL).	SELECT
CUSTID or RECORDER	Text	customer-id (recorder-id)	SELECT, WHERE
CHANNEL	Numeric	Channel (stratum)	SELECT, WHERE
START STARTTIME	Date	Start-time	SELECT, WHERE
DTSTAMP CHNLCUTTIMESTAMP	Date	Timestamp	WHERE
STOP STOPTIME	Date	Stop-time	WHERE

Field	Type	Description	Clause
MSTART STARTREADING	Numeric	Meter-start Reading	WHERE
MSTOP STOPREADING	Numeric	Meter-stop Reading	WHERE
MMULT METERMULTIPLIER	Numeric	Meter Multiplier	WHERE
MOFFSET METEROFFSET	Numeric	Meter Offset	WHERE
PMULT PULSEMULTIPLIER	Numeric	Pulse Multiplier	WHERE
POFFSET PULSEOFFSET	Numeric	Pulse Offset	WHERE
POP POPULATION	Numeric	Population	WHERE
WEIGHT	Numeric	Weight	WHERE
SPI	Numeric	Seconds Per Interval	WHERE
UOM UOMCODE	Numeric	Unit of Measure Code	WHERE
TIMEZONE	Numeric	Timezone	WHERE
DESC DESCRIPTOR	Text	Descriptor	WHERE
TZSN TZSTDNAME	Text	Timezone standard name	WHERE
EDITBYRS	Boolean	Edited By Rules Language flag	WHERE
EDITED	Boolean	Edit flag	WHERE
INTVALID INTERNALVALIDATION	Boolean	Interval Validation Flag	WHERE
EXTVALID EXTERNALVALIDATION	Boolean	External Validation Flag	WHERE
MERGE MERGEFLAG	Boolean	Merge Flag	WHERE
ARCHIVE DELETEFLAG	Boolean	Archive Flag	WHERE
ORIGIN	M,S,P	Origin Flag	WHERE
VALFLAGE	E or blank	Validation Flag (Energy Discrepancy)	WHERE
VALFLAGI	I or blank	Validation Flag (Number of Intervals)	WHERE
VALFLAGO	O or blank	Validation Flag (Outage)	WHERE

Field	Type	Description	Clause
VALFLAGN	N or blank	Validation Flag (Non-normal)	WHERE
TKWRIT	Boolean	TK Written Flag	WHERE
DST	Y or N	DST Participant Flag	WHERE
INTERVALCOUNT	Numeric	Interval Count	WHERE
DCFLOW	D	DC Flow	WHERE
VMSG MESSAGETEXT	Text	Validation Messages	WHERE
TRAIL EDITTEXT	Text	Edit Trails	WHERE

Comparison Operators

The following table lists the valid comparison operators that can be used in embedded SQL queries:

Comparison Operator	Description
=	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

LIKE	Wild-card comparison. The wild-card patterns that you can use are:
	• % : match any string of any length (including zero length)
	• _ : match on a single character
	Example:
	`CUSTID LIKE 'RES%'`
	`CUSTID LIKE 'RES10_'`

NOT LIKE	Inverse of LIKE.

Comparison Operator	Description

IN	Equal to a list of values. Lists must be enclosed in parentheses and comma separated.
	Example:
	`CHANNEL IN (1,2)`
	For regions where comma is used as decimal digit, numeric values must be enclosed in single quotes.
	Example:
	`METER_MULTIPLIER IN ('1,02', '2,01')`

NOT IN	Inverse of IN.

Supported Programs

The following Load Analysis programs support embedded SQL queries:

In CTL files:

- X210 Cut Series Validation
- X410 Load Data Reporter
- X440 Summary Reporter
- X470 Late Cut Report
- X480 Validation Statistics Reporter
- X490 Cut Series gap Reporter
- X530 Cut Series Overlap reporter
- X720 Direct Output (CLDB)
- Y240 Load Data Extraction
- Y720 Direct Output (ELDB)
- Q91C Copy Cuts
- Q91D Delete Cuts
- Q91M Move Cuts

In KYS files:

- X810 Key Generator

Custom Variables

Custom Key Generator variables can also be used in embedded SQLs. They can be referenced just like any other variable with certain limitations (see below).

Example:

```
@SELECT FKEY WHERE RATE = 'N1';
```

See **oula_server.cfg.xml** on page 1-14 in the *Oracle Utilities Load Analysis Configuration Guide* for more information about configuring custom Key Generator variables.

Limitations

1. Certain operators cannot work with custom variables. The operator must be one of "=", "!=", " LIKE " and " NOT LIKE ". For example, the following embedded SQL would be invalid:

   ```
   @SELECT FKEY WHERE RATE IN ('R1', 'N1');
   ```

   ```
   @SELECT FKEY WHERE RATE > '01';
   ```

2. The length of the compare value cannot EXCEED the length of custom variable. For example, if a custom field is defined with a length of 2, compare value cannot exceed 2 characters in length. For example, the following SQL would be invalid since RATE is defined with LENGTH=2:

   ```
   @SELECT FKEY WHERE RATE = 'N2024';
   ```

3. Multiple wild cards (%) are not allowed. Custom fields do not work with more than one wild card (%) in the compare value. For example, the following embedded SQL would be invalid:

   ```
   @SELECT FKEY WHERE RATE LIKE '%N%';
   ```

Chapter 5

Setting Up Your Oracle Utilities Load Analysis System

This chapter explains how to set up input files and establish procedures necessary for the successful operation of the Oracle Utilities Load Analysis System. If Oracle Utilities Load Analysis is already installed and working at your utility, you may wish to skim this chapter. On the other hand, if you are responsible for setting up a new system or maintaining an existing one, you should read the following pages carefully.

This chapter covers the following topics:

- **Creating Common Required Input Files**
- **Guidelines for Establishing Customer-ids and Channel-numbers**
- **Establishing Editing Procedures**

If you are implementing a new Oracle Utilities Load Analysis System, you may need to tailor a group of special input files to your utility's specific requirements. Once you have established these files, they will rarely change. You will also want to establish, up front, certain conventions and procedures as guidelines for personnel working with the system. This chapter provides useful information for both of these initial activities.

Creating Common Required Input Files

Oracle Utilities Load Analysis includes input files with default values. You should examine these files carefully, and make any changes necessary to have them conform to the needs of your operation. These files should require little if any modification once they are established:

- **How to Create the Validation Environment Files (TGX21B.ENV)**
- **Multiple Sets of Validation Criteria**
- **How to Create the Holiday File (TGY31C.HOL)**
- **How to Create the Time-Of-Use File (TGY31D.TOU)**
- **How to Create the Season File (TGY31E.SEA)**

How to Create the Validation Environment Files (TGX21B.ENV)

The Validation Environment File is very important, because it establishes tolerance levels for the internal and external validation tests that Oracle Utilities Load Analysis performs during the Data Validation process. "Internal" validation tests check the quality of data by comparing data within a cut; "external" validation tests compare data between two cuts in a series. The internal validation tests are: the Number of Intervals Test, the Energy Discrepancy Test, the Uncorrected Power Outages Test, the Non-Normal Intervals Test, the Zero Intervals Test, the High and Low Interval Demand Tests, and the Spike and Dip Intervals Tests. The external validation tests are: the Recording Period Match-Up Test and the Meter Reading Match-Up Test.

You can think of the Validation Program as a screen or filter that allows only data of a certain quality to pass through into the ALDB. The Validation Environment File allows you to control the "density," or selectivity, of the "filter." Tighter tolerances may mean less data, but of a higher quality; while looser tolerances usually mean more data, with less precision.

Typically, these tolerance levels are set at system installation by personnel at your utility. You may wish to create multiple sets of validation criteria for different kinds of data — for example, one for billing data and one for survey data. The default Validation Environment File can be set up to apply different validation criteria to selected cuts during the Input process, using special records in the AXDB. The default file may be located in the COMMON\DATA directory located on the server.

One approach for setting up the Validation Environment File is to use the default values supplied with the system for a time, then evaluate where modifications are required. ***Note:*** If you do change the tolerance limits after a time, and it is desired that *all cut series be validated using the same tolerance criteria, it is advisable to re-validate the entire CLDB* using the OPTIMAL VALIDATION no review setting.

Once the tolerance levels have been established, you create the Validation Environment File with the fourteen commands shown in Figure below.

BLOck *n*

DATe [*start–time*] [*stop–time*]

ENErgy [**OFF** | *e1,e2* | 0.98,1.02]

MULtiplier [*n.n* | 1.0]

OUTage [*k* | 0]

NONnormal [*n* | 0]

ZERo [*m* | *n*% | **OFF**]

HIGh *n.n* [*i* | 0]

LOW *n.n* [*i* | 0]

SPIke [*n* | 0] [*p*% | 50%]

DIP [*n* | 0] [*p*% | 50%]

TIMe [[*mm1*][*:ss1*],[*mm2*][*:ss2*] | 60,15]

METer [*m1,m2* | 1,1]

REPort [**CUTs** | **SERies**]

WARning DES1 *name start length . . .***DES5** *name start length* **STAtus** *status-codes*

EXEmpt [*ValidationTest UOM1 UOM2 ... UOMn*]

STA [*status code list*]

NNS [*Non-normal status codes*]

Validation Environment File Format

When creating the Validation Environment File, you must enter each command on a separate line. You need enter only the first three letters of the command. You may use commas or blanks to separate elements of the command. If you do not specify a command, Oracle Utilities Load Analysis will automatically use the default value (underlined).

> *Note: Each test is summarized below; for additional details, see Chapter 10: Re-Testing Data Quality Using the Cut Series Validation Program (X210, X220)*

DATe [*start–time*] [*stop–time*]

- Date — Use this optional command to limit the dates of cuts to be validated, where "start-time" specifies the beginning boundary of the date range and "stop-time" specifies the ending boundary of the date range. Those cuts in series requested in the Validation Control File whose start times are within the date range will be validated. If you omit the hh:mm:ss from the start-time, it defaults to 00:00:00; if you omit the hh:mm:ss from the stop-time, it defaults to 23:59:59. If you specify a start-time but no stop-time, the stop-time defaults to infinity. If you supply neither time, the DATe Command has no effect on processing.

ENErgy [**OFF** | *e1,e2* | 0.98,1.02]

- Energy — Use this command to establish tolerance levels for the Energy Discrepancy Test. Specifically, enter values for "e1, e2" to specify the range in which the ratio of metered energy to interval energy in each cut must fall. The valid range is 0 to 1.999. The default range is 0.98 to 1.02 (i.e., a ±2% discrepancy).

  Entering "OFF" in the Energy Command will turn off the Energy Discrepancy Test. *However*, this should be done *only* if your utility does not use metered energy in Oracle Utilities Load Analysis, or if testing for an energy discrepancy is otherwise detrimental to the validation process; *not* just because there is a large difference between your metered energy and interval energy.

MULtiplier [*n.n* | 1.0]

- Multiplier — Use this command to establish the allowable number of meter multiplier differences per cut for the Energy Discrepancy Test. The default is 1 times the meter multiplier.

OUTage [*k* | 0] [CON]

- Outage — Use this command to establish tolerance levels for the Uncorrected Power Outages Test. The value you supply for "k" specifies the maximum allowable number of intervals with status codes of "1" (uncorrected outage) per cut. The default is 0 intervals. If the CON (or CONSECUTIVE) keyword is present, the validation will be based on the maximum number of *consecutive* outages. If not provided, then validation will be based on the *total* number of outages.

NONnormal [*n* | 0] [CON]

- Non-normal — Use this command to establish tolerance levels for the Non-normal Intervals Test. The value you supply for "n" indicates the maximum allowable number of intervals with non-normal status codes (i.e., values from "2" to "9") per cut. The default is 0 intervals. If the CON (or CONSECUTIVE) keyword is present, the validation will be based on the maximum number of *consecutive* non-normal intervals. If not provided, then validation will be based on the *total* number of normal intervals.

ZERo [*m* | *n*% | **OFF**]

- Zero — Use this command to specify the maximum number of intervals in a cut whose data value can be 0 with status code blank (" ") or "A" through "I". This command will accept either a constant or a percentage of the total number of intervals in setting this limit. If the command is omitted or if the keyword 'OFF' is coded, no zeros-testing will be performed.

SPIke [*n* | 0] [*p*% | 50%]

- Spike — Use this command to establish tolerances for the Spike Intervals Test. The value you supply for "n" indicates the number of highest values or peaks to be averaged from the cut. Each interval in the cut is compared to this average to determine if it is abnormally high (a spike). "n" must be an integer from 1 to 10; the default value of "n" is 0. The value you supply for "p%" indicates a percentage by which an interval's value is not to exceed the average of the peaks. "p%" must be an integer from 1 to 100; the default for "p" is 50%. (Note that the '%' symbol is optional for the "p" parameter.) If you omit this command, the Spike Intervals Test will not be performed.

DIP [*n* | 0] [*p*% | 50%]

- Dip — Use this command to establish tolerances for the Dip Intervals Test. The value you supply for "n" indicates the number of intervals to compute a rolling average over for the test. Each interval will be compared to the rolling average of the "n" intervals preceding it to determine if it is abnormally low (a dip). "n" must be an integer from 1 to 10; the default value of "n" is 0. The value you supply for "p%" indicates a percentage by which an interval's value is not to exceed the computed rolling average. "p%" must be an integer from 1 to 100; the default for "p" is 50%. (Note that the '%' symbol is optional for the "p" parameter.) If you omit this command, the Dip Intervals Test will not be performed.

HIGh *n.n* [*i* | 0]

- High — Use this command to establish tolerances for the High Interval Demand Test. The value you supply for "n.n" (a required parameter) indicates the level above which an interval's demand will be considered high; "i" (optional) specifies the maximum allowable number of consecutive intervals with demand above "n.n". The default for "i" is 0 intervals. If you omit this command, the High Interval Demand Test will not be performed.

LOW *n.n* [*i* | 0]

- Low — Use this command to establish tolerances for the Low Interval Demand Test. The value you supply for "n.n" (a required parameter) indicates the level below which an interval's demand will be considered low; "i" (optional) specifies the maximum allowable number of consecutive intervals with demand below "n.n". The default for "i" is 0 intervals. If you omit this command, the Low Interval Demand Test will not be performed.

  Note: *The High and Low Interval Demand Tests check interval demands, not raw data values as stored in the cut. This fact should be considered when choosing the high or low limits designated by "n.n".*

TIMe [[*mm1*][*:ss1*],[*mm2*][*:ss2*] | 60,15]

- Time — Use this command to establish tolerance levels for the Recording Period Match-Up Test. The tolerance levels may be specified in minutes, seconds, or a combination of the two. If seconds are specified, they must be immediately preceded by a colon (:), whether or not minutes are also specified. The value you supply for "mm1:ss1" specifies the maximum time gap (also called "underlap") allowed between two cuts; "mm2:ss2" specifies the maximum overlap between two cuts. The default gap is 3600 seconds (60 minutes); the default overlap is 900 seconds (15 minutes).

METer [*m1,m2* | 1,1]

- Meter — Use this command to specify tolerance levels for the Meter Reading Match-Up Test. The value you supply for "m1" specifies the maximum gap (or "underlap") allowed between two meter readings; "m2" specifies the maximum overlap. The default gap is 1 meter reading unit; the default overlap is also 1 meter reading unit.
- Offsets — As of Client/Server Oracle Utilities Load Analysis Release 8.0, the Offsets Command is no longer applicable, but will be accepted as comments.

REPort [**CUTs** | **SERies**]

- Report — Use this command to determine if the entire cut series containing an invalid cut will be reported (by the Load Data Reporter step), or if the report will be restricted to invalid cuts. Enter "CUT" to report only invalid cuts. Enter "SERIES" to report the entire cut series containing an invalid cut. The default is "SERIES".

WARning DES1 *name start length . . .***DES5** *name start length* **STAtus** *status-codes*

- Warning — Use this command to cause the program to print a warning message in the Execution Log below each cut whose meter multiplier differs from the corresponding value in the previous cut of the same cut series. In addition, the DES subcommand prints a warning message if any of up to five specified fields in the descriptor differ from that of the previous cut in the same cut series. Each of up to five fields in the descriptor may be designated for "warning testing" by repeating the specification DES name start length the desired number of times. The STAtus subcommand prints a warning message if the status code of any interval matches those specified in the command.

  In each descriptor field specification, **DES** is a required keyword that indicates the beginning of a new specification; **name** is the name by which the field will be referenced in the Execution Log; **start** is the starting position of the field within the descriptor; and **length** is the length in bytes of the field. As many WARning commands as needed are allowed, but the DES keyword, name, start, and length pertaining to each individual descriptor field must all be on the same line.

  To specify status codes for the **STA**tus subcommand, include a non-delimited list of status codes for which warnings are to be printed. Only Oracle Utilities Load Analysis status codes are reported by this function. See Status Codes on page 3-5. For example, the following command produces a warning message for cuts containing status codes Q, L, and 9:

```
WAR STA QL9
```

If the WARning Command is coded with no DES or STAtus specifications, only differences in meter multiplier will be cited in warning messages.

Note: Because floating point numbers can be subject to precision irregularities, if the difference between two meter multipliers is less than .01% of the larger, the two values will be considered to be equal and will not be cited in a warning message.

STA [*status code list*]

- **STAtus** — User this command to scan all status codes of a cut, and cause the cut to fail the validation if a status code of a specified type (or types) is encountered.

  Example: STA QTSY

  The above example will flag all cuts being validated that have status codes "Q", "T", "S" or "Y" as invalid and write appropriate validation messages in the validation messages portion of the cut.

  The program will report in the environment report the new command and the values set. In addition, the program will report in the Execution Log that the cut has failed the STA validation test in a new column labeled "S" which will be next to the "HI/LO %". The failures will be noted in the body of the report with an "S" for each cut failure.

  An example Validation Message is:

  ```
  INVALID STATUS(ES) FOUND STARTING AT INTERVAL: 1 7 25 121 217.
  ```

EXEmpt [*ValidationTest UOM1 UOM2 ... UOMn*]

- **EXEmpt**— Use this command to exempt cuts with specified units-of-measure from undergoing specified Validation tests. This command can be repeated multiple times in a single environment file. Each instance of this command allows the user to specify a Validation test and a list of UOM codes that are to be exempted from that test, leaving the test applicable for cuts of any other UOM code. Multiple EXEmpt commands may be specified for any test if all the UOMs (separated by either commas or blanks) that are to be exempted from it will not fit on a single command. The UOM codes exempted from each test will be shown in the Environment Report on a line or lines immediately following the description of the test. Additionally, a new Environment Report page ("Interpretation Of Exempted Unit Of Measure Codes") will be produced showing, in numeric order, all the numeric UOM codes that have been exempted from any Validation test, together with descriptive names (KWH, KVARH, etc.) for those that are documented Oracle Utilities Load Analysis UOM codes.

  The EXEmpt command uses the following test keywords:

  MET-UNDER—Exemption from the Meter Underlap element of the Meter Reading Match-Up test.

  MET-OVER—Exemption from the Meter Overlap element of the Meter Reading Match-Up test

  ENErgy—Exemption from the Energy Discrepancy test (comparing metered to interval energy)

  TIM-UNDER (or **TIME-UNDER**)—Exemption from the Time Underlap element of the Recording Period Match-Up test

  TIM-OVER or (**TIME-OVER**)—Exemption from the Time Overlap element of the Recording Period Match-Up test

  OUTage—Exemption from the Uncorrected Power Outages test

  NONnormal—Exemption from the Non-normal Intervals test

  HIGh—Exemption from the High Interval Demand test

LOW—Exemption from the Low Interval Demand test

SPIke—Exemption from the Spike Intervals test

DIP—Exemption from the Dip Intervals test

ZERo—Exemption from the Maximum Consecutive Zero Intervals test

UOM Codes: A comma- or space-separated list of unit-of-measure codes to be exempted from the specified Validation test. Only those entered UOM codes that exist in the cuts being validated will have any actual effect on the running of the Validation program, and only documented Oracle Utilities Load Analysis UOM codes will have descriptions shown in the Environment Report Interpretation Of Exempted Unit Of Measure Codes.

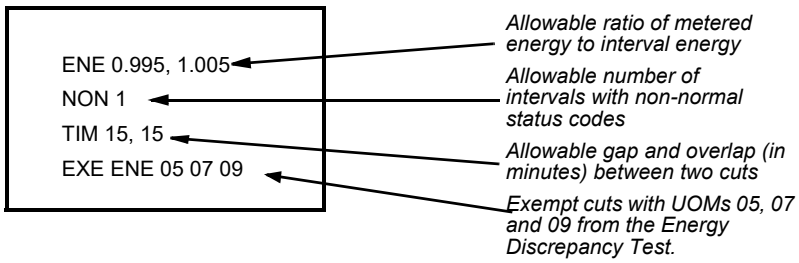


Figure 5-1 Sample Validation Environment File

In this example, the analyst set new tolerance levels for the Energy Discrepancy Test, Non-Normal Intervals Test, and the Recording Period Match-Up Test. No values were entered for the Outage, Meter, Multiplier, High, or Low commands, so Oracle Utilities Load Analysis will use the system default values for the Uncorrected Power Outages Test, the Meter Reading Match-Up Test, and the meter multiplier portion of the Energy Discrepancy Test; the High and Low Interval Demand Tests will not be performed. In addition, cuts of UOMs 05, 07, and 09 will be exempted from the Energy Discrepancy Test.

NNS [*Non-normal status codes*]

- **NNS**: A non-delimited list of status codes that are to be considered non-normal for purposes of the validation. The list may contain any character of the following set:

  023456789ABCDEFGHIJKMNOPQRSTUVWXYZ

Multiple Sets of Validation Criteria

In addition to overall tolerances, you may also use the Validation Environment File to define multiple sets of validation criteria for use with particular cut series or rate classes. Begin each of these extra sets of tolerance commands with a BLOck Command:

BLOck n

The BLOck Command indicates that all tolerance commands entered after it, until the next BLOck Command, if any, are to be considered part of validation criteria set "n", where "n" is an integer from 1 to 999. You may enter up to 999 BLOck commands, followed by their respective tolerance commands; the block numbers must be in ascending sequence, but gaps are permitted. The tolerance commands used in each individual Block are the same as those used to define the overall validation criteria just discussed.

If you use a BLOck Command in the Validation Environment File, then for each cut series to be validated, the Validation Program will search the Auxiliary Database (AXDB) for a Validation (type V) record with that series' key. If such a record exists, and the number in its text field matches the "n" on any Validation Environment File BLOck command, the tolerance commands in that Block will be used to validate that cut series. If a cut series has no type V record in the AXDB, or has one that specifies a number other than one defined on a BLOck Command, it will be validated according to the overall tolerance commands (specified before the first BLOck Command). (***Note:*** The AXDB is described in *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180).*)

If the first Environment File command *is* a BLOck Command, the Validation Program's defaults (underlined in Figure) will be used as the overall validation criteria, because no tolerance commands have been entered that are not part of a special Block.

You may also set up validation environment files to e used with specific databases (see *Oracle Utilities Load Analysis Configuration Guide*).

How to Create the Holiday File (TGY31C.HOL)

The Holiday File (.HOL) is a list of the dates of all national and local holidays that apply to your service territory. This file is required for analysis and reporting programs, because holidays are normally associated with different load patterns. Typically, this file is created at system setup, and added to each year. You should keep past holidays in the file, because the analysis and reporting programs will require the historical information. The default file is located in the Common/Data directory. The file's name and location are defined in the server configuration file CSLSTAR.GLB.

You create this file using the following record format:

mm/dd/yy

Enter one date per line. The length of the file is unlimited.

Figure 5-2 Figure 5-3 is an example of a Holiday File.

```
01/01/97
02/16/97
04/17/97
05/25/97
07/04/97
09/07/97
11/26/97
12/25/97
01/01/98
02/15/98
05/24/98
07/04/98
09/06/98
11/11/98
11/24/98
12/25/98
01/01/99
01/16/99
02/20/99
05/29/99
07/04/99
09/04/99
11/11/99
11/23/99
12/25/99
```

Figure 5-3 Sample Holiday File

How to Create the Time-Of-Use File (TGY31D.TOU)

The Time-Of-Use (.TOU) File tells Oracle Utilities Load Analysis how to evaluate the interval data when producing reports and analyses. Specifically, you use TOU schedules to identify which time periods your utility considers on-peak, off-peak, and shoulder (in-between) for each day of the week and holidays.

You can set up nine different TOU schedules in the TOU File, and each schedule is limited to nine periods.

Each TOU schedule *must* classify every hour of every day in the week and holidays. There can be no overlaps or gaps between periods.

When constructing the TOU File, you define each period with a single record, using the following format. You must follow the defined order and separate each parameter with one or more blanks.

schedule period day-list time-range [comment]

- Schedule — Enter a number to identify the TOU Schedule you are creating. The first schedule you enter in the file must be assigned a "1", **and all succeeding schedules must be numbered consecutively. There is no set code to indicate schedule type, but the schedule numbers must be in order.**
- Period — Enter a number to identify the period you are defining in the record. The first period you enter in the schedule must be assigned a "1", and all succeeding periods in the schedule must be numbered consecutively. There is no set code to indicate period type. However, it is important to be consistent when assigning period numbers to period types. For example, if Period 1 is designated as ON PEAK, then Period 1 should be used as ON PEAK periods for all schedules.
- Day-list — Enter a number to identify each day of the week to which the period applies. You must use the following codes:

  1 - Sunday

2 - Monday

3 - Tuesday

4 - Wednesday

5 - Thursday

6 - Friday

7 - Saturday

8 - Holiday.

You can enter from one to eight day codes in a record. Separate your entries by commas only — blanks are not allowed. You must enter the codes in ascending order.

- Time-range — Enter the hour, minute, and second when the period begins and the hour, minute, and second when the period ends. Remember: 24:00:00 is not allowed in Oracle Utilities Load Analysis. Note also: this range must not cross over midnight.
- Comments — Optional. Enter notes for your reference. Only the last defined comment per period will be used on reports.

A sample TOU File is shown in Figure 5-4.

```
1  1  2,3,4,5,6      08:00:00  20:59:59   ON PEAK
1  2  2,3,4,5,6      00:00:00  07:59:59   OFF PEAK
1  2  2,3,4,5,6      21:00:00  23:59:59   OFF PEAK
1  2  1,7,8          00:00:00  23:59:59   OFF PEAK

2  1  2,3,4,5,6      12:00:00  19:59:59   ON PEAK
2  2  2,3,4,5,6      08:00:00  11:59:59   SHOULDER
2  2  2,3,4,5,6      20:00:00  23:59:59   SHOULDER
2  3  2,3,4,5,6      00:00:00  07:59:59   OFF PEAK
2  3  1,7,8          00:00:00  23:59:59   OFF PEAK

3  1  2,3,4,5,6      10:30:00  14:29:59   ON PEAK
3  2  2,3,4,5,6      08:00:00  10:29:59   SHOULDER
3  2  2,3,4,5,6      14:30:00  20:59:59   SHOULDER
3  2  7              08:00:00  20:59:59   SHOULDER
3  3  2,3,4,5,6,7    00:00:00  07:59:59   OFF PEAK
3  3  2,3,4,5,6,7    21:00:00  23:59:59   OFF PEAK
3  3  1,8            00:00:00  23:59:59   OFF PEAK
```

Figure 5-4 Sample Time-Of-Use File

This file contains three different schedules. Look at the first schedule. For Monday through Friday, 8:00:00 a.m. through 8:59:59 p.m. (20:59:59) is “on-peak”. From midnight (00:00:00) through 7:59:59 a.m., and 9:00:00 p.m. (21:00:00) through midnight (23:59:59) are “off-peak”. All day Saturday, Sunday, and holidays are classified “off-peak”. Notice how every second in the week is accounted for.

How to Create the Season File (TGY31E.SEA)

The Season File can be used to apply different Time-of-Use schedules to different portions (seasons) of a reporting or analysis period. For example, on peak and off peak hours may be defined differently for summer and winter months, or a third, shoulder period defined for summer only.

The Season File consists of one or more season schedules. All records for each season schedule are grouped together, in ascending order by season schedule number. Each record specifies a time period, assigns to it a season, and associates with it a Time-Of-Use schedule. A maximum of 12 distinct seasons may be defined in each season schedule.

Many separate time periods may be defined as belonging to the same season (winter in several different years, for example). Within any season schedule, every time period assigned to the same season must also be associated with the same time-of-use schedule.

Each Season File record has the following format:

Season-sch# season# tou-sch# start stop [season-name]

where:

season-sched#	is a positive integer identifying the season schedule to which this record belongs.
season#	is a number from 1 to 12 identifying a season.
tou sched#	is a non-negative integer identifying the time-of-use schedule (defined in the Time-Of-Use File) to be used over this season.
start	is the start time of this season segment, in "MM/DD/YY-hh:mm:ss" format. This must be the beginning of a full day; if a time (hh:mm:ss) other than 00:00:00 is specified, it is ignored and the program proceeds using a time of 00:00:00.
stop	is the stop time of this season segment, in "MM/DD/YY-hh:mm:ss" format. This must be the end of a full day; if a time (hh:mm:ss) other than 23:59:59 is specified, it is ignored and the program proceeds using a time of 23:59:59.
season-name	is a description of the season. A given season-number must be paired with the same season-name throughout a single season-sched#.

A Season Schedule must contain season segments covering all times in the period being reported or analyzed, with no gaps and no overlaps. If the program detects a gap or an overlap in the designated Season Schedule, it will produce an error message and stop processing. Also, all Time-of-Use schedules referenced by the designated Season Schedule over the analysis period must be defined in the Time-Of-Use File. TOU Period numbers defined in all Time-of-Use schedules referenced must be consistent. For example, if TOU Period 1 is defined as "ON PEAK", then TOU Period 1 should be "ON PEAK" for all TOU schedules involved.

Figure 5-5 illustrates a Season File containing two season schedules, one defining two distinct seasons and the other defining four. Either Season Schedule one or Season Schedule two could be

used for the period from July through December of 1992, for example; but only Season Schedule two could be used over the same months in 1993.

```
1,1,1,11/15/91-00:00:00,04/20/92-23:59:59,WINTER
1,2,2,04/21/92-00:00:00,06/11/92-23:59:59,SPRING
1,3,3,06/12/92-00:00:00,08/31/92-23:59:59,SUMMER
1,4,4,09/01/92-00:00:00,11/14/92-23:59:59,FALL
1,1,1,11/15/92-00:00:00,04/20/93-23:59:59,WINTER
1,2,2,04/21/93-00:00:00,06/11/93-23:59:59,SPRING
1,3,3,06/12/93-00:00:00,08/31/93-23:59:59,SUMMER
1,4,4,09/01/93-00:00:00,11/14/93-23:59:59,FALL


2,1,1,11/12/91-00:00:00,04/20/92-23:59:59,WINTER
2,2,2,04/21/92-00:00:00,11/11/92-23:59:59,NON-WINTER
2,1,1,11/12/92-00:00:00,04/20/93-23:59:59,WINTER
2,2,2,04/21/93-00:00:00,11/11/93-23:59:59,NON-WINTER
2,1,1,11/12/93-00:00:00,04/20/94-23:59:59,WINTER
```

Figure 5-5 Sample Season File

This file contains two different schedules. Look closely at the first schedule. Four distinct seasons are defined and associated with different Time-of-Use schedules. Notice how all times in the analysis period are accounted for.

Guidelines for Establishing Customer-ids and Channel-numbers

Other factors you must take into account when setting up your Oracle Utilities Load Analysis System are customer-ids and channel-numbers. Here are some guidelines you should consider:

- Each customer/channel number combination **must** have a unique identifier.

Customer-ids

- A customer-id can be 1 to 64 characters long. You ***can*** use any combination of characters (A - Z) and digits (0 - 9). Avoid using special characters (anything that is not a letter or number) because they have special meanings in Oracle Utilities Load Analysis.
- Because it is possible to retrieve customer data by querying on any position(s) in the customer-id field, it is useful to include meaningful information in the customer-id. For instance, you might wish to incorporate rate codes, strata numbers, and/or SIC codes. Further, because many Oracle Utilities Load Analysis reports are automatically sorted into ascending customer-id order, it is a good idea to use the first few characters as a group code.
- Because personnel will often have to input customer-ids (especially during editing and reporting), it is a good idea to keep the customer-id as short as practical. This will help minimize data entry time and possible errors.

Channel-numbers

- A channel-number must be an integer from 0 through 32767.
- It is a good idea to devise a consistent scheme for assigning channel numbers for multi-channel devices. Typically, within a group code, each number is consistently assigned to a specific type of end-use measurement. For example, for residential customers, in an end-use sample, channel 1 might be assigned to total kilowatts, channel 2 to refrigerators, channel 3 to water heaters, etc. For large industrial customers, channel 1 might be assigned to total kilowatts, channel 2 to KVAR, channel 3 to temperature, etc.

  Note: You will usually assign customer-ids and channel-numbers to the recording devices as they are installed. However, if for some reason you need to change the IDs, you can do so during data input using the AXDB (see Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)).

Establishing Editing Procedures

It is important to establish a consistent set of rules for editing data. Because many individuals are typically involved in this process, a well-defined and clearly communicated set of procedures is necessary to help ensure a consistent result.

For example, given the nature of interval recording devices and translators, you may find that certain problems occur consistently. There are typically numerous solutions and/or workarounds for any given problem. It is a good idea to issue a list of such frequently occurring problems and their preferred solution to everyone involved in editing.

The Automatic Editor enables you to specify a set of Edit Rules for handling particular data problems. These rules indicate the conditions to be corrected and the actions to be taken. You supply Edit Rules to the Validation Program (X210) in a special input file. If this Edit Rules File is present, the Validation Program determines whether the specified rules are applicable to each cut that it processes. If a rule is found to apply to a given cut, the Validation Program creates one or more Load Data Editor (X310) commands to correct the problem diagnosed. Actual editing is performed by a subsequent Oracle Utilities Load Analysis Editor step, after which the edited cut is put through Validation again.

For details, see the Automatic Editor section of *Chapter 10: Re-Testing Data Quality Using the Cut Series Validation Program (X210, X220)*.

Chapter 6

Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs

If you have diligently read the previous chapters, you understand the "big picture" of how Oracle Utilities Load Analysis works, and you recognize the key terms and definitions associated with the subsystem. Now it is time to look closely at some of the most commonly used procedures in the LDMS.

This chapter describes the *primary ways of* getting data into the CLDB Interval Database — the Production Input and Direct Input programs. You use these programs when you want to input new load data from an Interval Data source into Oracle Utilities Load Analysis. Both the Input and the Load Data programs input data to the CLDB, and also automatically initiate Load Data Validation and Reporting for you.

The primary methods for inputting data into the CLDB are the Production and Direct Input Procedures. These procedures can handle large volumes of data, and both incorporate extensive automatic data validation, optional editing, and reporting. These processes may optionally be automated by setting up a Sequencer. (See *Oracle Utilities Load Analysis User's Guide*.)

This chapter covers the following topics:

- **Which Procedure Should You Use?**
- **Overview of the Production Input, Direct Input, and Load Data Input Procedures**
- **Steps for Using the Production Input, Direct Input, or Load Data Input Procedures**

Which Procedure Should You Use?

The Direct Input and Production Input Procedures are identical. Your choice of programs will depend upon the location at which your load data currently exists:

- **Production Input and Direct Input —** Use the Direct Input Program when the data already exists in Oracle Utilities format. (The Oracle Utilities formats are described in detail in *Appendix B: Record Formats Used by Oracle Utilities Load Analysis Input Programs*.) Your translation software may deliver the data in a Oracle Utilities format, or your data processing department may convert it for you. Direct Input will load files that are stored locally on your Client machine. Production Input loads data that is stored on the Server. **Note:** There are two Oracle Utilities input formats. The Standard Input format (.INP) and the Enhanced Input format (.LSE).

Overview of the Production Input, Direct Input, and Load Data Input Procedures

Both procedures include a Data Input Program, the Load Data Validation Program, and the Load Data Reporter (Figure 6-1). You supply the data files with the cuts to be entered and submit the procedure. Oracle Utilities Load Analysis automatically runs the programs and creates the Control files that are required for input to the Validation and Reporting programs.

The **Input Programs** enter the cuts into the CLDB. The **Validation Program** evaluates the quality of the load data in the cuts according to the standards established in the Validation Environment File (TGX21B.ENV) and flags any cuts that fail. The Validation Program performs two types of tests: internal and external. Internal tests evaluate the quality of data within a cut; external tests match each cut to the preceding and following cuts in the series. By using BLOCK commands in the Validation Environment File in conjunction with the AXDB, you can apply multiple sets of validation criteria to selected sets of incoming cuts. The **Load Data Reporter** issues descriptive validation messages for "bad" cuts, so that you can correct any problems in the editing phase.

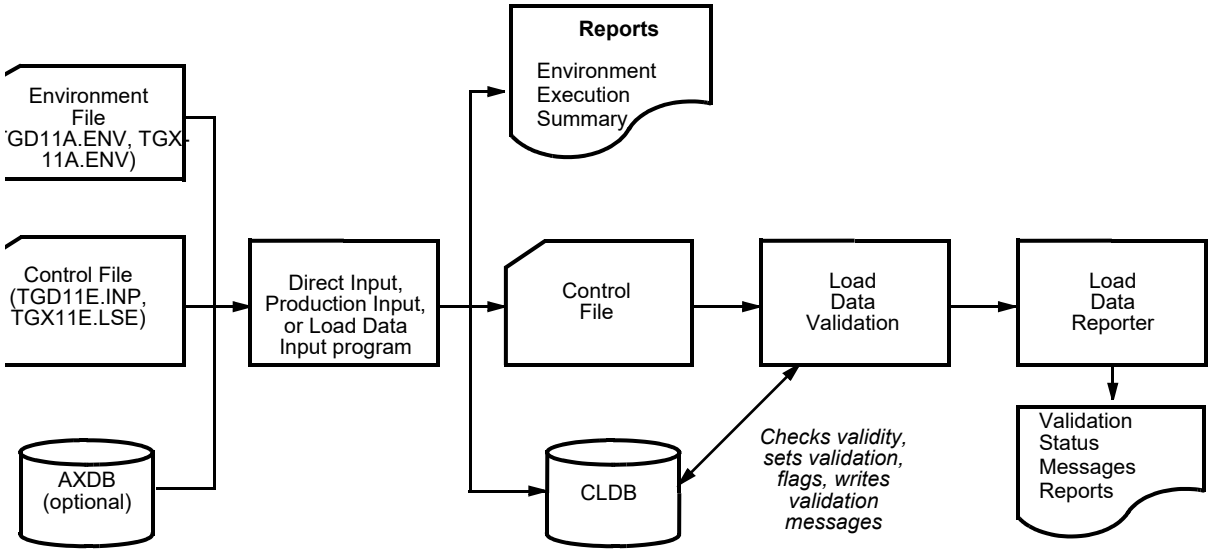


Figure 6-1 Input Procedure

Steps for Using the Production Input, Direct Input, or Load Data Input Procedures

Here is a brief list of the steps you will follow when entering data into the CLDB, using either the Production or Direct Input. The remainder of this chapter explains these steps in detail.

Summary — Using Direct Input or Load Data Input Procedures

1. Gather translated data files for input.
2. Determine whether or not data requires special modification using the Auxiliary Database (AXDB). If data requires modification, update or select the appropriate AXDB before proceeding.
3. Create Input Environment File (TG11B.ENV).
4. Submit Job.
5. Check output reports.

Step 1: Gather translated data files for input (.LSE or .INP)

These files will become the Input Control or Load Data File. Remember, these files must be in the format that corresponds to your Input Program. If your input files have the .LSE extension, you must use X110 (see page 6-18 for more information about using X110). The Direct Input and Production Input programs require records to be in the standard Oracle Utilities Formats .LSE or .INP (see **Appendix B**: **Record Formats Used by Oracle Utilities Load Analysis Input Programs**).

Step 2: Determine whether data requires special modification via the Auxiliary Database (AXDB)

The AXDB enables you to select or modify incoming cuts before they are input to the CLDB. Specifically, you can:

- Translate an old customer-id or channel-number to a new one.
- Modify a descriptor field.
- Select specific cuts from a group of cuts in the Input Control File.
- Adjust start- and stop-times.
- Apply different validation criteria to selected incoming cuts.

If any one or more of the above is true, you must include instructions for the modifications in the Auxiliary Database before running the Input Program. Turn to **Chapter 7**: **Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)** and follow the instructions for updating the Auxiliary Database before continuing with this procedure.

If you want to enter data into the CLDB without AXDB modification, skip this step.

Step 3: Create the Environment File (TGX11B.ENV)

You use the Environment File to set the conditions under which you want the incoming data to be processed. The file consists of three commands (followed by sub-commands), as shown in the following box.

LOAd [**IDLength** *nn*] [**KEY**] [**REPlace**] [**TRAnslatedOnly**] [**VALid**]

FULIintervals [**CODe** | **ADJ**] [**DESc** | **NODesc**]

DST [**ADJusted** | **NO** | **YES**]

UOM *From–UOM–code To–UOM–code*

NONzero *s* [**LIST** | **NOList**] [**BYPass** | **WRIte**]

ENHanced [**WRIte** | **NOWrite**]

UNDercount [**FILl** | **REJect** | **ASis**]

OVErcount [**REJect** | **ASIs**]

LOG [**ERRors**] [**REJects**]

STAtus *from-status to-status*

IDAppend <*original ID start pos*>, <*num chars*>, <*exception-1*>..., <e*xception-n*>

PREfix prefix

SUFfix suffix

SPLit YEArly

AGG [**m** | **0**]

TZW [<*TZWOGMT*> <*target TZSN*> | <*TZSN*>]

LOAD — Use this command to establish general operating conditions for the processing of incoming data. The "Load Command" has the following options:

REPLACE — In the REPLACE mode, an incoming cut will automatically replace an existing cut in the CLDB that has the same cut key. *Use this option with care.*

If you do not specify REPlace, any incoming cuts with keys that already exist in the CLDB will not be entered, and the existing cuts in the database will remain intact.

KEY — Use this option when you want to input only specific cuts from the incoming file. This option limits processing to those incoming cuts whose Select keys appear in the AXDB (see *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)*).

If you do not use the KEY option, all incoming cuts will be processed.

VALID — Use the VALid option to set the internal validation, external validation, merge, and archive flags on incoming data to "Yes" — in other words, to override Oracle Utilities Load Analysis's built-in validation procedure. If you do not use VALid, all flags will be set to "No", and Oracle Utilities Load Analysis will apply the validation tests to the cuts.

You would typically use the VALid option only when meter readings are not available for validation testing, or when you are loading historical data that was previously validated using a set of criteria other than that currently in use. You would not want to re-validate the historical data (in other words, change its quality), because previous analyses were probably run using the old criteria.

If you are entering new data, you should not use this option, unless you are loading an ELDB directly.

IDLENGTH — Use this command to shorten the incoming customer-id to the specified length ("nn").

TRAnslatedOnly— Use this command to load only translated cuts (those with a corresponding "T" record in the AXDB). Cuts with no corresponding AXDB "T" record will no be written to the database, and are written into an output file in the job directory called "IDS_NOT_TRANSLATED.CTL." This file can be modified and used to append the AXDB update control file.

FULIintervals [**CODe** | **ADJ**] [**DESc** | **NODesc**]

FULLINTERVALS — Use this command to adjust start- and stop-times of cuts to full-interval boundaries (e.g., start-time 13:00:00, stop-time 23:59:59). This command is used in conjunction with descriptor/time adjustment records (type D) in the AXDB (see *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)*). The code parameter is an optional 3-character code that identifies which cuts should be adjusted, as indicated on the AXDB. The default code is "ADJ". The "DESC" option indicates that the description from the AXDB 'D' record is to be used in the CLDB. The "NODESC" option causes the description text to be ignored.

Special note about using FULLINTERVALS to adjust start-and stop-times

The AXDB must contain a descriptor/time adjustment record for each cut to be adjusted by FULLINTERVALS. See *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)* for a discussion of the AXDB.

DST [**ADJusted** | **NO** | **YES**]

DST — This option indicates to the system what DST format the incoming data is recorded in. DST Y (default) indicates the incoming data is DST participant (23 hours in spring DST, 25 intervals in fall DST). DST N indicates the incoming data does NOT participate in DST (Always 24 hour days). DST A indicates that the incoming data participates in DST, but is adjusted to 24 hour days. (A missing interval inserted for spring DST, and the extra interval is combined for fall DST). (This command should only be used with .INP files.)

UOM *From–UOM–code To–UOM–code*

UOM — Use this option to translate *from* the foreign unit of measure codes contained in the direct input files *to* the Oracle Utilities unit of measure codes as input data is being loaded. Use of this option will not alter the original data, only the unit of measure code in the cut being created.

From-UOM-code — This is the foreign unit of measure code that, if contained in the direct input data, will be replaced in the resultant cut by the specified Oracle Utilities unit of measure code.

To-UOM-code — This is the Oracle Utilities unit of measure code to be stored into the cut instead of the foreign UOM code in the original input data.

NONzero *s* [**LIST** | **NOList**] [**BYPass** | **WRIte**]

NONZERO — An interval status code of '9' is used to indicate a *missing interval* in a Oracle Utilities Load Analysis database. Missing intervals by definition will have a value of 0 and a status of '9'. *Oracle Utilities Load Analysis will not store an interval with a status of '9' and a value other than 0* in a database. If the Direct Input program encounters such an interval in an .inp file, its default action is to store a status of '9' and a value of 0 in the corresponding database record; the number of intervals modified this way is displayed in the Execution Log.

With the NONZERO command, you specify a status code to be substituted for the '9' in such intervals, and indicate that the interval values are not to be zeroed. The command has one required and two optional parameters.

s — Use this required parameter to specify the alternate status code to be substituted for a '9' in an input interval with a nonzero value. This must be a single, unquoted character in the range J - Z or 0 - 9. Any interval that is input with a status code of '9' and a nonzero value will be written to the database with the '9' replaced by this alternate status code. The number of nonzero status '9' intervals found in each input block will be displayed in the Execution Log.

LIST — If you specify this optional parameter, the data record sort# and relative interval number of each interval found with a status code of '9' and a nonzero value will be listed in the execution log. The default is NOLIST.

BYPASS — By specifying this option, you tell the program to bypass writing a cut to the database if its .inp block contains one or more intervals with a status of '9' and a nonzero value. In effect, you are defining the nonzero-status '9' combination as an error. Use this option in combination with the LIST option if you want to locate such intervals for correction before entering the cut into the database. The default is to WRITE the cut to the database.

ENHanced [**WRIte** | **NOWrite**]

ENHanced— This command determines whether X110 writes the cuts that do not conform to the Oracle Utilities format that was the standard before Release 8.0. These cuts are accessible by Oracle Utilities Load Analysis Release 8.0 and later, but cannot be accessed by Oracle Utilities Load Analysis programs in releases before 8.0. In the latter case, however, they can still be used by other Oracle Corporation products, such as the Oracle Utilities Energy Information Platform.

ENHanced has the following options:

WRITE — Default. In the WRITE mode, cuts that do not conform to the pre-Release 8.0 Oracle Utilities standard format will be written to the database. Each non-standard cut written to the database will be noted in the Execution Log.

NOWRITE — In the NOWRITE mode, X110 will not write the Enhanced Format cuts to the database, and will flag them as errors in the Execution Log.

Note: If using the **WRITE** format of the **ENHanced** option, the keys of all enhanced format cuts written to the database, can, at the user's option, be listed in a file named WarnFile.XXX, located in your job directory, where XXX is the value of REPTMASK (.dat in most instances) in the CSLSTAR.GLB (Run Time) file. If REPTMASK has not been set, the default extension for this file is .REP (WarnFile.REP). The WarnFile.XXX is created if CSWARN is set to YES in the CSLSTAR.GLB file and is not created otherwise. See *Oracle Utilities Load Analysis Configuration Guide* for information about editing the CSLSTAR.GLB file.

The table below summarizes the results of the ENHanced settings

CSWARN setting in CSLSTAR.GLB	ENHanced /WRITE/ NOWRITE command in Environment file	Results
YES	ENHanced WRITE	WarnFile.XXX created and cut written to database

CSWARN setting in CSLSTAR.GLB	ENHanced /WRITE/ NOWRITE command in Environment file	Results
YES	ENHanced NOWRITE	WarnFile.XXX not created and cut not written to database
YES	Omit ENHanced WRITE / NOWRITE	WarnFile.XXX created and cut written to database
NO	ENHanced WRITE	WarnFile.XXX not created and cut written to database
NO	ENHanced NOWRITE	WarnFile.XXX not created and cut not written to database
NO	Omit ENHanced WRITE / NOWRITE	WarnFile.XXX not created cut and written to database

UNDercount [**FILl** | **REJect** | **ASis**]

UNDercount— This command dictates program behavior when the number of intervals supplied is fewer than the number of intervals expected.

UNDercount has the following options:

FILl — Default. If the number of intervals supplied is fewer than the number of intervals expected the cut will be filled out with missings. A warning message to this effect will be posted to the Execution Log.

REJect — If the number of intervals supplied is fewer than the number of intervals expected the cut will be rejected as an error.

ASIs — If the number of intervals supplied is fewer than the number of intervals expected the cut will be accepted as is. A warning message to this effect will be posted to the Execution Log.

OVErcount [**REJect** | **ASIs**]

OVErcount— This command dictates program behavior when the number of intervals supplied exceeds the number of intervals expected.

OVErcount has the following options:

REJect — Default. If the number of intervals supplied is greater than the number of intervals expected the cut will be rejected as an error.

ASIs — If the number of intervals supplied is greater than the number of intervals expected the cut will be accepted as is. A warning message to this effect will be posted to the Execution Log.

LOG [**ERRors**] [**REJects**]

LOG - Use this command to enable file logging options to track cuts that failed input ("not entered") for any reason. This command has the following options:

ERRors - Use this option to enable logging of keys that failed input due to error(s) in the incoming data file. The list of keys generated will be stored in a log file called "DAT_ERRORS.LOG". If there are no errors in the incoming data file, the log file will not be created.

REJects - Use this option to enable logging of keys that failed input because they were rejected by the database for any reason. The list of keys generated will be stored in a log file called "REJECTS.LOG". If there are no keys rejected, the log file will not be created.

STAtus *from-status to-status*

STAtus - Use this command to change all occurrences of a specified status code in the input file (*from-status*) to a target status code (*to-status*) when the data is saved to the database.

IDAppend <*original ID start pos*>, <*num chars*>, <*exception-1*>..., <e*xception-n*>

IDAppend - Use this command to append a portion of the original Recorder ID onto the end of a translated ID (from the AXDB), thereby retaining some of its original ID. This command takes the following parameters:

<original ID start pos> is the beginning of the string in the original ID to be appended to the translated ID.

<num chars> is the number of characters from the start position to append to the translated ID

<exception-1> ... **<exception-n>** dictate which string(s) combinations are exempt from this operation and will NOT be appended if found. Strings listed must be of the same length as the <num chars> parameter.

Note: if the translated ID is not found for a specified cut, the IDAppend operation is not performed for that cut.

AGG [m | 0]

AGG - Use this command to aggregate all incoming cuts to specified SPI.
Example AGG 1800 will convert all incoming cuts from any SPI to 1800. (30 minutes)

PREfix SUFfix

PREfix / SUFfix - Use this command to add a prefix / suffix onto the recorder ID of all incoming cuts. Ensure that the combination of prefix, recorder ID and suffix is not longer than 64 characters or truncation may occur.

SPLit YEArly

SPLit YEArly - Use this command to split incoming cuts on full year boundaries. For example, all cuts that span 01/01/01 - 00:00:00 will be split into two cuts with the first starting on the original start time and the second starting at 01/01/01 - 00:00:00.

TZW [<*TZWOGMT*> <*target TZSN*> | <*TZSN*>]

TZW - Use this command to assign Time Zone Standard Names (TZSN) to incoming cuts. There are two modes available: Mapping and Set All. *Mapping* mode maps specified Time Zones West of GMT values to a target TZSN. The environment file can contains multiple mappings. *Set All* mode sets the time zones of all cuts to the specified TZSN.

Step 4: Run the Input Procedure (X110, X111)

Once you have chosen all of the necessary inputs on the Submit screen — the Environment File, the Load Data File, and (optionally) the auxiliary database — you are ready to run the Input Procedure.

Processing

The following paragraphs describe the order in which the Input procedures process the data. If you find that errors have occurred, or if you have general questions about what the programs do, you should find this description helpful.

Each input block of header records and data records used to create a single CLDB cut is read. It is examined for errors and, if any are found, the block is discarded, diagnostic messages are written, and the next block of data is read. When running the program, each block may contain no more than 9000 interval records, and if it contains from 3001 to 9000 data records (i.e., more than 36,000 intervals), the program will split the resulting cut into separate consecutive cuts; each of these but the last will have 36,000 intervals, with the last containing the remaining intervals from the input block. If there are no errors in the block, the information in the header and data records is used to build a CLDB cut. This involves the assignments of descriptive fields from the corresponding information on the first four header records (see *Appendix B: Record Formats Used by Oracle Utilities Load Analysis Input Programs*). In addition, each interval and status code value is assigned to an interval in the cut in the order they are read in. If Direct Input doesn't recognize a status code, it changes it to an 'X'.

The program checks the AXDB for records associated with the incoming cut. If the IDLength option of the Load Command is used, the incoming customer-id is truncated to the length indicated; in that case, the customer-id on an AXDB must only equal the truncated customer-id for a match to occur. If "S", "D", and "T" records exist for the cut, processing will occur in the following order: 1) selection will occur if LOAD KEY was specified; 2) descriptors and/or start and stop times will be modified, and 3) translation will be performed on the incoming customer-id and channel. If "T" records are present in the AXDB, for the current cut, all other keys for AXDB functions will be based on the translated Recorder ID and channel. After all AXDB processing has occurred, Oracle Utilities Load Analysis will translate characters such as "&" and "@" to "-".

If the Interval Data is in Standard (.INP) format, the program checks the alternate format flag, element 8 of the First Header Record, before assigning the cut's pulse multiplier. If the alternate format flag's value is "0", the pulse multiplier is assigned from element 5 of the Second Header Record; the recorded value should be greater than (0.99999). If the alternate format flag's value is "1", the alternate pulse multiplier is assigned from element 3 of the Third Header Record; the recorded value should be less than (1.00000). If the pulse multiplier is zero, the block is discarded, a diagnostic message is written, and the next block of data is read.

If the VALID option of the LOAD Command is specified, then internal and external validation flags are set to "YES" by the standard Direct Input Program (X110). This type of processing is done on a cut by cut basis by setting the Validate Record Flag to 'N'.) This is normally done when meter readings aren't available for the Direct Input Control File and no validation testing can be performed on the cut. If the meter readings are available the VALID option shouldn't be specified. In this case, the Direct Input Program will set the validation flags to "NO". Either way, the Merge and Archive flags are set to the same values as the validation flags.

Start- and stop-times may be adjusted depending on the DST option of the LOAD Command. The DST option is normally specified if the incoming start- and stop- times are indicated in standard time, even during the daylight savings time period. If the DST option is specified, then all incoming start-times and stop-times that fall within the DST period are adjusted by one hour so they conform to local times. This applies to INP format only.

> **Note**: To avoid problems when processing requests and inputs around the DST time changes, the programs do not allow input times of 02:00:00 through 02:59:59 on the sprint DST day in a DST system.

Next, the Load Data Validation Program examines an entire CLDB cut series (*up to 10,000 cuts for each cut that was input*). The Input Program generates a file of keys for the cut series to be validated. If multiple cuts in a series are entered, redundant cut series keys may be written to this file for validation. To prevent redundant validation processing, the Validation Program is preceded by a sort of the Control File so that redundant cut series keys are discarded automatically. Optimal Validation may also produce a "double validation" effect. This behavior is expected.

Validation processing proceeds in ascending time order for each cut series. Each cut in the series is read and tested for internal validation. Next, each cut is compared to the following cut for external validation. Finally, all the cuts are rewritten to the CLDB with appropriate values stored in the internal and external validation flags, and the validation message areas.

The Load Data Reporter is invoked immediately following Validation, and reports all cuts/or cut series that were validated and contain invalid data. If there are no invalid cuts, the Load Data Reporter Control File will be empty, and "No valid requests" will be printed on the Load Data Execution Log.

Chapter 7

Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)

At times, when using the Direct Input procedures, you will want to select or modify cuts from the input files before entering them into the CLDB. The Auxiliary Database (AXDB) enables you to do this.

This section includes the following topics:

- **What Does the Auxiliary Database Do?**
- **Which Programs Control the Auxiliary Database?**
- **Steps for Using the AXDB Update and Summary Reporting Programs**

What Does the Auxiliary Database Do?

The function of the Auxiliary Database (AXDB) is to give you greater flexibility in modifying incoming data before input to the CLDB — specifically for cut selection, ID translation, descriptor modification, start- and stop-time adjustment, and the application of different validation criteria. If you wish to modify incoming data in any of these ways, you must prepare the AXDB, by using the AXDB Update Program, before running either the Direct Input procedures, or you may set up multiple AXDBs, each for a specific purpose.

You can think of the AXDB as a place where you keep instructions for modifying incoming records (Figure 7-1).

Essentially, the AXDB contains a list of cut keys and associated modifications. Before entering data into the CLDB, the Input programs compare each incoming cut to the list of cuts in the AXDB, and if any of them match, automatically modifies them according to the information in the AXDB.

You make up this "AXDB list" (the contents of the AXDB) using four types of records. Each type of record makes a different kind of modification to the incoming data. The four types of records are:

- **D** — DescriptorUsed to modify the descriptor and/or adjust the start- and stop-time of an incoming cut.
- **S** — Selection Used to select specific cuts from an input file.
- **T** — TranslationUsed to modify an incoming customer-id
- **V** — ValidationUsed to select a set of validation criteria for a cut series.

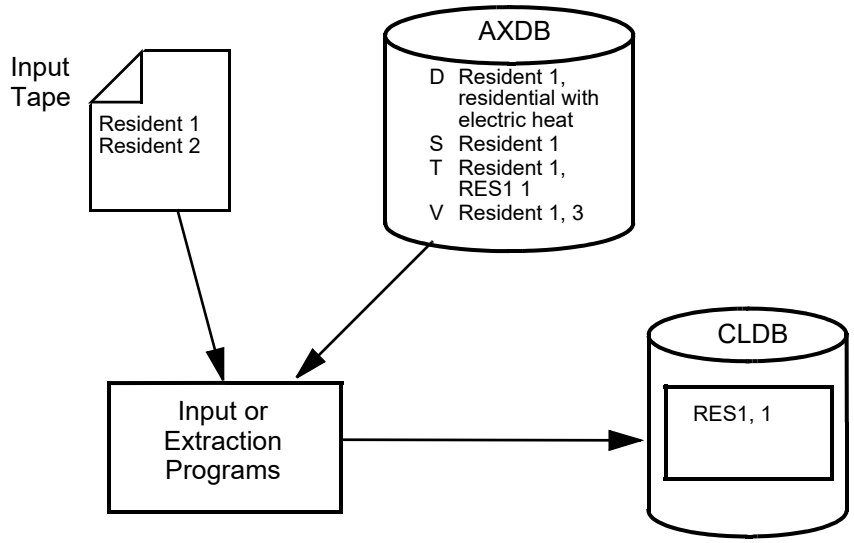


Figure 7-1 The AXDB: Used to Select and/or Modify Cuts for Input to the CLDB

Records of each type remain in the AXDB until you change or delete them. In this way you do not have to constantly re-enter often-used records.

Special Note about the Auxiliary Database

Every time you run either one of the Input programs, Oracle Utilities Load Analysis automatically checks the incoming data against all of the D and T records in the AXDB. Therefore, before running the Input programs with the AXDB, you must verify that any pre-existing D and T records will make the modifications you desire. If not, use the AXDB Update Program to modify or delete them. S records are activated only if you select KEY in the Input Environment File. V records are activated only if you include a BLOCK Command in the Validation Environment File (see *How to Create the Validation Environment Files (TGX21B.ENV)* on page 5-2).

Which Programs Control the Auxiliary Database?

Three programs are used to create, update, and report the AXDB:

- **AXDB Update (X180)** — used to re-create or update specific record types in the AXDB — in other words, to specify your instructions to the programs for selecting and/or modifying incoming records.
- **AXDB Summary Reporting (X170)** — used to report the contents of the AXDB. You would typically run this program in preparation for running the Direct Input or Load Data Input programs.

These programs are described on the following pages.

Steps for Using the AXDB Update and Summary Reporting Programs

Once you have determined that you want to update the contents of the AXDB in preparation for modifying incoming data, use the following procedure:

Summary — Updating the AXDB

1. Create the AXDB Update Environment File (TGX18B.ENV) — the D, S, T, and/or V records you specify for modifying incoming cuts.
2. Create the AXDB Control File (TGX18A.CTL).
3. Run the Auxiliary Database Update Program (X180).
   Note: You must perform the three steps above for *each record type* (D, S, T, V) you wish to modify. For example, if you want to modify D and T records, you must run the job twice.
4. Create the AXDB Reporting Environment File (TGX17B.ENV), and run the AXDB Reporting Program (X170) to ensure that the desired records were properly entered.

After following the above steps, you are ready to input data to the CLDB using the Direct Input or Load Data Input programs.

These steps are described in detail on the following pages.

> **Note:** Because the AXDB Update program can be used to delete and replace existing records in the database, you should retain copies of your control files in case you need to re-enter the deleted records.

Step 1: Create the AXDB Update Environment File (TGX18B.ENV)

The Environment File specifies the record type you want to enter into the AXDB (in other words, the contents of the Control File). The Environment File Command uses this format:

RECord {**D** | **S** | **T** | **V**}

DATe

MODe {ADD| DEL | REP} **MODe** defines the mode in which the program will operation. This must be followed by one of the sub commands.

MOD DEL [ALL] This is the default setting. This command deletes records in the database identified by records in the control file. Partial keys (custid, custid & channel) will be accepted. To delete all records of a specified type dictated by the type command, add the ALL keyword (ex: MOD DEL ALL). Reports will display a "DELETED FROM DATABASE" status in the report under the text column.

MOD ADD This command adds new records of the specified rectype to the database. Any existing records will not be replaced.

MOD REP This command provides the ability to replace and add records to the database that are supplied in the control file.

- **The Record Command** is required. Your options are:

  D — Descriptor Use this record type when you want to modify the descriptor field of a cut series or a specific cut, and/or adjust the start-and stop-time fields.

S — SelectionUse this option when you want to select specific cuts from the input file.

T — TranslationUse this when you want to change the customer-id and channel field.

V — ValidationUse this when you want to select a specific set validation criteria for a cut series. (See also the section on the Validation Environment File in *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*.)

Again, you can choose only one option at a time.

- **The Date Command** is required when start times are included in your Control File records. It tells the program that there are going to be start times in the AXDB Control File. It may not be used with the RECORD V Command, because type V records may not contain start times. The default is no date.

Step 2: Create the AXDB Update Control File (TGX18A.CTL)

The Control File consists of the records you want to place in the AXDB. Essentially, they consist of two parts:

- the keys identifying the cuts you wish to modify or select.
- the modifications.

Create these records using the following formats:

- **For S-type records (Selection)**

  customer-id, channel [,start-time]

  - *Customer-id* is always required. It is the current ID of the incoming record.
  - *Channel* is optional. Use it only when you want to select records with specific channel numbers.
  - *Start-time* is required only if you want to select records with specific start-times. Remember, if you include start-times in your AXDB Control File, you must select the DATE option in the AXDB Environment File.

An example of a Control File corresponding to type "S" records is:

```
10001 /* KEYS FOR SELECTION
20001 /* SELECT ALL DATES
30001,1
30001,3
```

The above Control File specifies which cuts are to be selected for entry into the CLDB by the Load Data Input Program. These include all cuts with customer-ids of 10001 or 20001, and cuts with customer-id 30001 and channel numbers 1 or 3.

An example of a Control File (with an Environment File containing the commands "RECORD S" and "DATE") is:

```
10001,   11/18/97-09:00:00    /* KEYS FOR SELECTION
20001,1,12/19/97-11:45:00     /* SPECIFIED DATES ONLY
```

The preceding Control File specifies that incoming data with customer-id 10001 and start-time 11/18/97-09:00:00 or cuts with customer-id 20001, channel number 1 and start-time 12/19/97-11:45:00 are to be selected.

- **For T-type records (Translation)**

  customer-id, channel [,start-time] ,'text-field'

  - *Customer-id* is always required. It is the current ID of the incoming record.

- *Channel* is optional. Use it only when you want to change the IDs on cuts with specific channel numbers.
- *Start-time* is required only if you want to change the IDs on cuts with specific start-times. Remember, if you include start-times in your AXDB Control File, you must select the DATE option in the AXDB Environment File.
- *Text field* is required. Enter the *new customer-id* for the cut. Enclose your entry in single quotes.

An example of a Control File (with an Environment File that specifies "RECORD T") is:

```
10001     A0001     /* KEYS FOR TRANSLATION
20001     B0001
30001,1   'C0001 1'
40001&    D0001
```

The above Control File provides a translation list to translate incoming cut customer-ids to Oracle Utilities Load Analysis customer-ids. All cuts with incoming customer-id 10001 will be translated to Oracle Utilities Load Analysis customer-id A0001 and all cuts with incoming customer-id 20001 will be translated to Oracle Utilities Load Analysis customer-id B0001. All incoming cuts with customer-id 30001 and channel 1 will have both customer-id and channel translated. The incoming channel must be specified, and the translated ID and channel must be enclosed in single quotes. All cuts with incoming customer-id 40001& will be translated to Oracle Utilities Load Analysis customer-id D0001.

- **For D-type records (Descriptor)**

customer-id, [channel] [,start-time] ,'text-field'

- *Customer-id* is always required. It is the current ID of the incoming record.
- *Channel* is optional. Use it only when you want to add or modify the descriptor on cuts with specific channel numbers.
- *Start-time* is required only if you want to add or modify descriptors on cuts with specific start-times. Remember, if you include start-times in your AXDB Control File, you must select the DATE option in the AXDB Environment File.
- *Text-field* — Enter the new descriptor for the cut. Enclose it in single quotes. You can enter up to 80 characters.

Special Note about using the D-type Records to adjust Start- and Stop-times

The "text field" may contain an optional three-character Time Adjustment code in positions 78 - 80 that works in conjunction with the Direct Input Environment File FULLINTERVALS Command (see *Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs*). If positions 78 - 80 of the text field contain the same three-character "code" as specified in the Direct Input FULLINTERVALS Command, the start- and stop-times of the cut are adjusted to full interval boundaries (e.g., start-time 13:00:00, stop-time 23:59:59). In this case, the use of the descriptor text to replace the incoming descriptor is dependent on whether the NODESC option of the Direct Input FULLINTERVALS Command has been specified. If the text is used, positions 78 - 80 are ignored.

An example of a Control File (with an Environment File containing "RECORD D") is:

```
1001,0    '0016942 ABC Co.'    /* KEYS FOR TRANSLATION
2001,2    '0017823 XYZ Inc.'   /* MODIFICATION
3001      '0029376 John Doe Corp.'
```

The above Control File provides a descriptor list to modify incoming descriptors. All cuts with customer-id 1001 will have their descriptors modified to '0016942 ABC Co.,' and all cuts with customer-id 3001 will have their descriptors modified to '0029376 John Doe Corp'. Customer-id 2001 will have its descriptor modified to '0017823 XYZ Inc.' for channel 2 only.

An alternate example of a Control File (with an Environment File specifying "RECORD D") is:

```
1001,0     ADJ'0016942 ABC Co.'          2001,2
ADJ'0017823 XYZ Inc.'
3001
```

The above Control File provides a descriptor/time adjustment list to modify incoming descriptors and/or start- and stop-times. The text "ADJ" in positions 78 - 80 of the text field is a code to adjust start- and stop-times to full interval boundaries. *The Direct Input Program must be invoked with the FULLINTERVALS Command for this text to be interpreted as "adjustment requests." Otherwise, it will be interpreted as part of the descriptor.* The processing of these type "D" records by Direct Input is dependent on whether the NODESC option of FULLINTERVALS has been specified. If NODESC was invoked, all cuts with customer-ids 1001 and 3001 will have their start- and stop-times adjusted to full interval boundaries, but the incoming descriptors will not be modified. The descriptor for customer-id 2001 will be modified to '0017823 XYZ Inc.' for channel 2 only and times will ***not*** be adjusted. On the other hand, if DESC is invoked, all cuts with customer-id 1001 will have their descriptors modified to '0016942 ABC Co.' and their start- and stop-times will be adjusted, all cuts with customer-id 3001 will have their descriptors modified to blanks and their times will be adjusted, and the descriptor for customer-id 2001 will be modified to '0017823 XYZ Inc.' for channel 2 only and times will ***not*** be adjusted.

- **For V-type records (Validation criteria)**

  customer-id, [,channel] ,'text-field'

  - *Customer-id* is always required. It is the current ID of the incoming record.
  - *Channel* is optional. Use it to identify cuts with specific channel numbers.
  - *Text field* — Enter an integer from 1 to 999 inclusive, without quotes. When you run Load Data Validation against cuts in this series, that program will use this number to select the validation criteria to be used.

    Start-time is *not* accepted on this record.

SET Command

The SET command can be used in place of the 'text-field' value in AXDB control files to add or update only part of the customer-id / descriptor field using AXDB fields defined in the oula_server.cfg.xml configuration file. See **oula_server.cfg.xml** on page 1-14 in the *Oracle Utilities Load Analysis Configuration Guide* for more information about configuring custom AXDB fields.

Syntax:

```
SET(< axdb-field>=<value> )
```

where:

- **axdb-field**: Specifies the AXDB field name to update. Field must match name that is defined in the server configuration file (oula_server.cfg.xml).
- **value**: The value to which to update the record. The text provided should be less than or equal to the specified AXDB field length. Text that exceeds the specified limit will be cropped to fit.

Updating Multiple Fields

Multiple fields can be updated for a single record with additional X=Y assignments, separated by semi-colon (;).

Syntax:

```
SET( <field1>=<value1>; <field2>=<value2> )
```

Control File Syntax

The SET command replaces the standard 'text-field' field of the AXDB Update Control file. As such, the valid control file syntaxes for the SET command are as follows:

```
customer-id SET(<field> = <value>)
customer-id channel SET(<field> = <value>)
customer-id channel start SET(<field> = <value>)
customer-id channel start SET(<field1> = <value1>;<field2> = <value2>)
```

T Records

The translation text for T-Type records can contain translation for customer-id AND channel. Because of this, AXDB field updates require special consideration.

AXDB Fields for T Records

Custom AXDB field definitions for T Records will always apply to the customer-id (you cannot create definitions for Channel.). For example, for the following entry:

```
<VARIABLE NAME="LOCODE"  RECTYPE="T" STARTPOS="8" LENGTH="2" />
```

The LOCODE field is defined as the 8th character position of the customer-id field, extending for 2 characters.

Special Fields

When updating T Records, the following additional special fields are recognized:

- **CUSTID**: This refers to the customer-id portion of the translation text. Refer to this field to update the entire customer-id.
- **CHANNEL:** This refers to the channel portion of the translation text. Refer to this field to update the channel only.

  Example:

  ```
  00990207 1 SET(CUSTID=RES1701ME; CHANNEL=1)
  ```

- **Blank Spaces:** Should updating T Records result in a customer-id field with leading blanks or blanks between characters, the blank spaces will be replaced with "_" characters.

  Example:

  Given LOCODE defined as follows:

  ```
  <VARIABLE NAME="LOCODE" RECTYPE="T" STARTPOS="8" LENGTH="2" />
  ```

  and the following AXDB Update CTL file:

  ```
  NEWCUT, 1 SET(LOCODE=MA)
  ```

  The resulting translated customer-id will be padded with underscores.

Step 3: Run the Auxiliary Database Update Program (X180)

Once you have entered the AXDB Environment File and the Control File, you are ready to modify the AXDB by running the Update Program. Use X180.

The AXDB Update Program will delete all AXDB records that have the record type you specified in the RECORD Command of the Environment File, and will then replace them with the new records in the AXDB Control File.

AXDB Update Processing (X180)

The first step in the AXDB Update Program may delete all records in the Auxiliary database with the record type specified in the RECORD Command of the Environment File. The program then updates Auxiliary Database records with that same record type using the Control File records. If DATE mode is in effect, the program requires Control File records to have a customer-id and start-time; otherwise, only a customer-id is required. Channel and text-field are always optional in the Control File.

The Load Data Input programs use the Auxiliary Database to assist in entering data in to the CLDB. When the Input Program runs in KEY mode, no cut is written to the CLDB unless there is an entry in the Auxiliary Database with a record type of "S" and a matching customer-id. If a channel and/or start-time were specified in the Auxiliary Database record, only those incoming cuts with the specified channel and/or start-time will be translated.

For T-type records, the first value in the text-field of the Auxiliary Database record is used as the translated customer-id, and the second value (if specified) is used as the translated channel. The incoming channel must be specified if a channel number is included in the text-field. Any further lookups in the AXDB will be based on this new ID.

If no time adjustment code is specified, a descriptor is automatically modified if there is a record in the Auxiliary Database with record type "D" and a matching customer-id. If a time adjustment code is specified, the start- and stop-times of the cut are adjusted to full interval boundaries (e.g., start-time 13:00:00, stop-time 22:59:59). If the NODESC option of Direct Input FULLINTERVALS has been specified the descriptor text will be ignored, allowing dates to be adjusted without overwriting the incoming descriptor. Otherwise, the descriptor is modified as usual. If a channel and/or start-time were specified in the Auxiliary Database record, only those incoming cuts with the specified channel and/or start-time will be processed. The entire value in the text-field is used to modify the descriptor unless positions 78-80 match the time adjustment code specified in Direct Input FULLINTERVALS.

Step 4: Run the AXDB Reporting Program (X170)

The AXDB Reporting Program will enable you to examine the contents of the AXDB. It is especially useful when you are preparing to execute the Direct Input or Load Data Input programs. Before running the AXDB Reporting Program, you must create an Environment File.

The Environment File is used to select the desired AXDB records for reporting. The Select Command has the following format:

SELect { D | S | T | V | ALL }

You may select any of the four record types, or "ALL":

- **D** reports all records in the descriptor/start- and stop-time modification table.
- **S** reports all records contained in the selection list.
- **T** reports all records in the translation table.
- **V** reports all records in the validation criteria table.
- **ALL** reports all records in the AXDB.

Once you have specified the AXDB and the Environment File, you are ready to run the AXDB Reporting Program. Use X170.

AXDB Reporting Processing (X170)

The AXDB Reporting Program locates and writes the desired records based on the record type specified in the SELECT Command.

AXDB Reporting Outputs: The AXDB Reporting Program will produce the following outputs, contained in the REPORT.HTML file:

- **AXDB Environment Report**
- **AXDB Execution Log**
- **AXDB Summary Report.**

Now that you have updated the AXDB, you are ready to input data. Turn to Step 3 in *Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs* to continue with the Input process.

Chapter 8

Manually Entering Data Into the CLDB (X120) and ELDB (Y220)

The Manual Entry Procedure is an alternate way of entering load data into the CLDB or ELDB. You would typically use this procedure when you want to enter small amounts of load data into the CLDB or ELDB, such as "system peak day demand" and "peak data weather," that is not in the two standard formats (.LSE or .INP)

You will use Manual Entry when you have small amounts of data to put into the CLDB or ELDB — in other words, when the Production Input, Direct Input, or Load Data Input programs would be too cumbersome. Because you are essentially typing data directly into the system, it is assumed that you will first verify its accuracy. For that reason, this procedure does not include automatic validation and extensive reporting (as do the Load Data Input and Direct Input procedures).

Figure 8-1 is an overview of the Manual Entry Procedure:

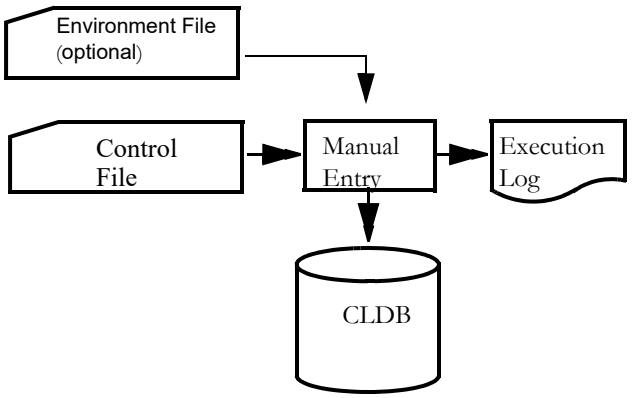


Figure 8-1 Manual Entry Flowchart

Steps for Using the Manual Entry Programs (X120 and Y220)

Here is a list of the steps you will follow to use these programs. Each of these steps is described in detail on the following pages.

Steps for Manually Entering Data into the CLDB or ELDB
1. Create Manual Entry Control File (TGX12A.CTL or TGY22A.CTL).
2. Create Manual Entry Environment File if desired (TGX12B.ENV or TGY22B.ENV).
3. Run Manual Entry Program (X120 or Y220).

Step 1: Create the Control File (TGX12A.CTL or TGY22A.CTL)

The Manual Entry Program builds cuts from a manually-created file (the Manual Entry Control File) and enters them into the CLDB or ELDB. The Control File you will create contains the cut key(s), interval data, (optionally) status codes, and other parameters.

You construct the Control File out of "blocks" of data, one block per cut. Use the following commands:

KEY *customer-id, channel, start-time*

SET UOM [*ui* | 01]

SET SECONDS-PER-INTERVAL [*spi* | 900]

SET POPULATION [*pop* | 0]

SET WEIGHT [*wt* | 0]

SET METER-MULT [*mm* | 0]

SET METER-OFFSET [*mo* | 0.0]

SET METER-START [*mstart* | 0.0]

SET METER-STOP [*mstop* | 0.0]

SET DES [*descriptor*]

SET DES1 [*descriptor1*]

SET DES2 [*descriptor2*]

DATa *Interval data*

STAtus *Status codes*

You can insert blank lines anywhere in the Control File, either within or between blocks. You can also insert commas or blanks between entries on a line. Commands must be entered in the following order: KEY, SET, DATA, STATUS.

- **Key** — Each block must begin with a KEY Command. Use it to establish the cut key — "customer-id, channel-number, start-time". ***Note:*** You can specify the start-time in either the "mm/dd/yy-hh:mm:ss" or "mmddyyhhmmss" format.
- **Set** — These optional commands can be used to override the default values for the cut's fixed data fields. (See *Chapter 3: The Oracle Utilities Load Data Format* for an explanation of these fields.) You can enter the SET commands in any order.

  PULSE-MULT — Enter a non-zero, positive real number. If a pulse multiplier is specified, all data values will be treated as pulses. If no pulse multiplier is specified, the data values will be treated as energy. Note that if no pulse multiplier is specified, "0" will be stored in the database; however, it is invalid to *specify* a pulse multiplier of "0". If a pulse offset is specified, a pulse multiplier must also be specified.

  PULSE-OFFSET — Enter a real number; positive, negative, or zero. The default is 0. If a pulse multiplier is specified, a pulse offset must also be specified.

  UOM — Enter the appropriate numeric code for the unit of measure. (See the *Quick Reference Guide* or *Appendix A: Oracle Utilities Unit of Measure Codes* for a list of Oracle Utilities UOM codes.) The default is 01, which stands for kilowatt hours.

  SECONDS-PER-INTERVAL — 60, 300, 900, 1800, 3600, or 86400. The default is 900.

  POPULATION — For use with statistics only. Enter a non-negative numeric value no larger than 99999999999. The default is 0.

  WEIGHT — For use with statistics only. Enter a non-negative numeric value no larger than 99999999999. The default is 0.

  METER-MULT — Enter a positive real number. If no value is specified, 0 will be stored in the database. While 0 is not a valid value for the meter multiplier, its presence in the database indicates that no meter multiplier was supplied.

METER-OFFSET — Enter a real number; positive, negative, or zero. The default is 0.

METER-START — Enter a non-negative real number. The default is 0.

METER-STOP — Enter a non-negative real number. The default is 0.

DES1 — Enter the first half (40 characters) of the cut descriptor.

DES2 — Enter the second half (40 characters) of the cut descriptor. Set DES2 is optional. If used, it must be accompanied by a Set DES1 Command.

DES — Enter the entire cut descriptor (80 characters). If DES is specified, then DES1 and DES2 should not be.

- **Data** — Use this required command to indicate the start of the load data. ***If a Pulse Multiplier has been provided,*** you must specify each value in integer form, preferably as a positive value less than 32767. If necessary, use the pulse multiplier and pulse offset fields to rescale and offset the data. If your data does include values that exceed 32767, Oracle Utilities Load Analysis will automatically rescale the data. ***If no Pulse Multiplier has been provided,*** data will be treated as energy. The data may be negative, and may include decimal digits.
- You can enter an unlimited number of load data records in the Control File.
- **Status** — This command is used to indicate the start of the status code records. Use of the status codes is optional, but if you do enter status codes, you must enter the Status Command. (***Note:*** You can find a list of status codes in Figure 3-2 in this manual, or in the *Quick Reference Guide*.)

  If you do not enter this command and the status codes, Oracle Utilities Load Analysis will automatically assign an 'A' status code to each interval ('A' means "normal, hand-entered"). If you do include status codes in the Control File, the number of status codes *must* equal the number of load data values. If not, the block will be discarded. ***Note:*** A status code of '' (normal data) **cannot** be used for manual entry input.

Figure 8-2 is an example of a Manual Entry Control File:

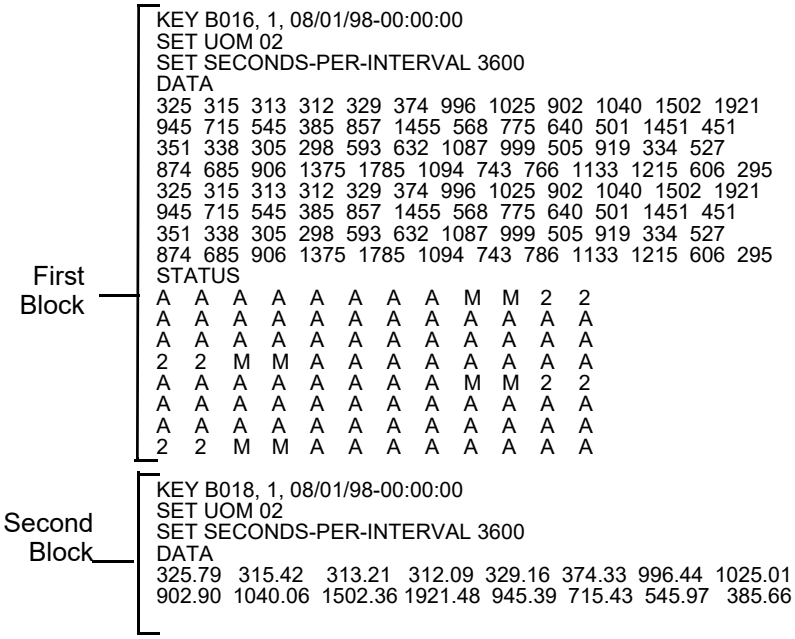


Figure 8-2 Manual Entry Control File

This manually-created file consists of two blocks of data, which Oracle Utilities Load Analysis will translate into two cuts. The cuts will have the same start-time, but will have different stop-times because they have an unequal number of interval values. Because the operator did not enter status codes for interval data in the second block, Oracle Utilities Load Analysis will automatically assign a status code of 'A' (normal — hand-entered) to each interval.

Step 2: Create the Environment File - Optional (TGX12B.ENV or TGY22B.ENV)

REPlace

The Environment File is optional, but if it is present and it contains the command REPlace, any cut created via this program will replace an existing cut with the same key. The default is not to replace cuts with identical keys.

Step 3: Run the Manual Entry Program (X120 or Y220)

Once you have constructed the Control File, and optionally the Environment File, you are ready to run the Manual Entry Program. Use X120 or Y220.

Manual Entry Processing

Each block of data is used to create a single cut. The stop-time is computed based on the start-time, the total number of data values, and the seconds-per-interval. If the STAtus Command is present, each load data interval is tagged with the corresponding status code provided. If no STAtus Command is present, then each load data interval is assigned the status code 'A'. The cut is flagged as internally and externally valid, and the Merge and Archive flags are set to "NO". The cut is then written out to the CLDB (or ELDB). Unless REPlace has been coded in the Environment File, *if a cut with the same key already exists in the CLDB (or ELDB), the block is discarded* and processing resumes with the next block. Any error in an input block will cause that block to be terminated and processing will continue with the next input block.

Chapter 9

Editing Data in the CLDB Using the Load Data Editor (X310 or X320)

After inputting data to the CLDB using the Load Data Input or Direct Input procedures, you will probably find that several cuts have failed your validation tests. The cuts should be corrected before they are transferred to the ALDB (archive database) or the ELDB (the working database for the Load Data Analysis Subsystem).

In this chapter, you will learn how to interpret Oracle Utilities Load Analysis's validation messages for "bad" cuts, and how to use the Load Data Editor Procedure to correct and re-validate the data.

Topics in this chapter include:

- **Steps for Using the Load Data Editor (X310 or X320)**
- **Load Data Reports and Validation Messages**
- **Summary of Edit Commands**
- **Guidelines for Using the Edit Commands**
- **Editor Command Descriptions**

The chapter concludes with a sample editing session.

You use the Load Data Editor (Figure 9-1) to modify cuts stored in the CLDB. You can change descriptive fields and interval data values, set storage flags, or synthesize new cuts from existing ones. You will create the edits using a powerful "edit command language."

Oracle Utilities Load Analysis gives you the option of "scanning" or checking your edits before actually applying them to the data. Once you have made your edits, the Editor automatically invokes the Validation Program to revalidate the edited cuts.

For each newly-edited cut, the editor automatically creates three records (unless you specify otherwise in the Environment File):

- **Inactive record** — the original unedited version
- **Active record** — the latest edited version
- **Edit trail** — a list of all modifications made to the data record.

In this way, you always have a copy of the original version.

Important Note about the Load Data Editor

The Editor stores the original cut and the *most recently edited version*. If you have edited a cut several times and wish to go back to an intermediate stage, you will have to undo your edits one by one until you get back to the desired version, or start over using the original cut.

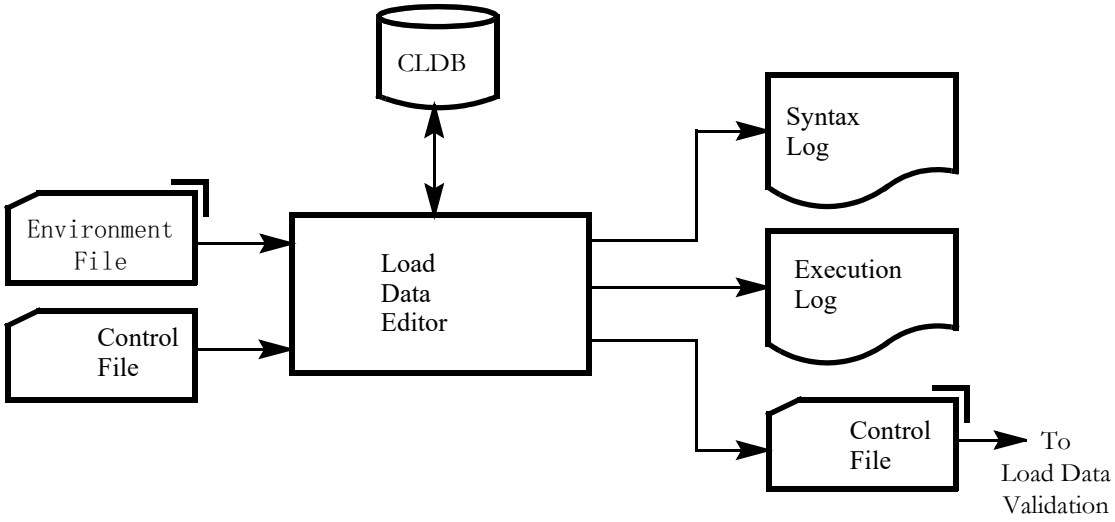


Figure 9-1 Load Data Editor

Steps for Using the Load Data Editor (X310 or X320)

Once you have determined that you need to modify data in the CLDB, follow these steps:

Summary — Using the Load Data Editor

1. Review validation reports for invalid data, or query using the Key Generator programs.
2. Determine type of edit to make for each error.
3. Create Control File containing edit commands (TGX31A.CTL).
4. Create Editor Environment File (TGX31B.ENV) — specifies whether or not the program checks your edits before you apply them to the database, and whether or not the program creates an edit trail and preserves a copy of the original version of edited cut.
5. Run Load Data Editor Program X310, or run X320, the Load Data Editor Syntax Scan, if you just want to check your edit commands in the Control File.
6. Check output reports.
7. Identify any remaining errors and return to Step 2 if necessary.

Note: This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Step 1: Review Validation Reports for Invalid Data

Oracle Utilities Load Analysis automatically produces a series of validation reports at the conclusion of the Input or Validation Procedure run. Examine these reports carefully to identify "invalid cuts" — cuts that failed validation testing and require modification before they can be archived into the ALDB or used for analysis. Three of these reports will be particularly useful to you at this stage:

- **Validation Environment File** — lists back the utility-specific criteria that Oracle Utilities Load Analysis used to evaluate the cuts. (This file was established at system setup. It is described in detail in *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*.)
- **Load Data Validation Program Execution Log** — a summary list of all cuts that were validated and their status.
- **Load Data Reports** — a detailed report on each invalid cut.

In some instances you may find that, in order to accurately diagnose the problem with a cut, you will need more information than is provided in the Load Data Report. Chapter 12 explains how to obtain reports that show every interval ("data dumps") and other detailed information.

The Load Data Validation Program Execution Log and the Load Data Reports are reviewed on the following pages.

Each column of the report is explained below:

- **Customer-id, Channel, Start-Date/Time** — The keys of all cuts entered during the last input procedure and their status. If there were any cuts already in the database from the same cut series as any newly-entered cuts, up to 200 of those cuts will appear on the report; two for each new cut. That is because the validation tests evaluate how new cuts match up with other members of their cut series.
- **Stop-Date/Time** — The recorded stop-date/time for the cut, from the translation file.
- **EDT** — Indicates whether or not the cut has been edited.

- **Unit of Measure** — Indicates the unit of measure associated with the cut's interval data.
- **Internal Validation** — Shows the results of the internal validation tests. (Internal validation tests check the data within a cut against standards established in the Validation Environment File.)
    - **NGY** — an E in this column means that the cut failed the "Energy Discrepancy Test" — the difference between the meter energy and the interval energy was greater than the acceptable meter multiplier differences, or was outside the allowed tolerance parameter range.
    - **INT** — an I in this column means that the cut failed the "Number of Intervals Test" — the number of data intervals in the cut did not correspond to the differences between the start and stop times.
    - **OUTAGE** — a number and percent in this column means that the cut contains outage intervals. However, a cut can have outages and still pass validation, as long as the number of outages is within the parameters established in the Validation Environment File.
    - **NONNORM** — a number and percent in this column means that the cut contains non-normal intervals (intervals with a status code of '2' - '9'). However, a cut can have non-normal intervals and still pass validation, as long as the number of non-normal intervals is within the parameters established in the Validation Environment File.
    - **ZERO** — a number and percent in this column means that the cut contains non-missing intervals with data value(s) of zero (0). However, a cut can have intervals with data values of zero and still pass validation, as long as the number of zero intervals is within the parameters established in the Validation Environment File.
    - **SPIKE** — a number and percent in this column means that the cut's interval data contains spikes (see *Chapter 10: Re-Testing Data Quality Using the Cut Series Validation Program (X210, X220)* for definition). However, a cut can have spikes in its interval data and still pass validation, as long as the number of spikes is within the parameters established in the Validation Environment File.
    - **DIP** — a number and percent in this column means that the cut's interval data contains dips (see *Chapter 10: Re-Testing Data Quality Using the Cut Series Validation Program (X210, X220)* for definition). However, a cut can have dips in its interval data and still pass validation, as long as the number of dips is within the parameters established in the Validation Environment File.
    - **HI/LO %** — a number before the slash in this column indicates the percentage of the cut's intervals whose demands failed the High Interval Demand Test; a number after the slash indicates the percentage that failed the Low Interval Demand Test. However, a cut can have high- and/or low-demand intervals and still pass validation, as long as the number of these intervals is within the parameters established in the Validation Environment File.
- **Status** — shows the internal and external validation status of the cut.
    - **IN** — a "V" in this column means that the cut passed all four internal validation tests; an "I" means that the cut failed one or more of the tests. The five columns to the left indicate which one(s) it failed.
    - **EX** — summarizes the results of the External Validation Tests. (External Validation Tests evaluate the match-up between a cut and the cut following it in the series according to standards established in the Validation Environment File.) A horizontal line in this column means that the cut is the last in its series, so Oracle Utilities Load Analysis cannot perform the test. The cut is considered "Invalid" but it does not necessarily require correction.
- **MRG and ARC** — The Merge and Archive flags indicate whether the cut is ready to be archived into the ALDB or merged into the ELDB (the on-line database used for analysis).

These flags are automatically set to NO by the Input and Editor programs, and normally left that way until it is time to archive the cuts to the ALDB. When you or another user attempts to move the cuts using the Scan, Archive/Delete Procedure, that procedure checks the Internal and External Validation Flags, and sets the Merge and Archive flags accordingly. (The procedure sets the Archive and Merge flags to YES only if the cut has been marked both internally and externally valid.) As explained later in this chapter, you can override this process by forcing the Merge and Archive flags to YES using the Load Data Editor before running the Archive Procedure, but this is not recommended.

Load Data Reports and Validation Messages

For each cut that it finds invalid, Oracle Utilities Load Analysis also automatically produces a Load Data Report. These reports provide more detailed information to help you evaluate problems. For example, the Load Data Validation Execution Log indicated that cut "B004, 1, 07/30/98-12:20:00" was internally invalid because it failed the Energy Discrepancy Test. The following Load Data Report for this cut shows that there is a large variance between metered energy and interval energy.

(INTERNAL) ENERGY DIFFERENCE (M-I): D	RATIO (M/I): R
where d = meter energy — interval energy	r = meter energy/interval energy

The cut has failed the **Energy Discrepancy Tests** — the difference between the meter energy and the interval energy is greater than the meter multiplier, and the ratio of the meter energy to the interval energy exceeds the allowable tolerances established in the Validation Environment File.

(INTERNAL) COMPUTED STOP TIME: mm/dd/yy-hh:mm:ss
where mm/dd/yy-hh:mm:ss is the computed stop-time based on the recorded start-time and the number of intervals in the cut.

The cut has failed the **Number of Intervals Test** — Oracle Utilities Load Analysis calculates how many intervals should exist in the cut by subtracting the cut's recorded start-time from its recorded stop-time, and converting the difference to interval equivalents (corrected for Daylight Savings Time, if applicable). If the expected number of intervals is not equal to the recorded intervals, the cut fails. Either the cut has too many or too few intervals, or one of the times is incorrect.

OUTAGES: k @ mm/dd/yy-hh:mm:ss
where k = number of consecutive intervals with outages, beginning with the interval whose recorded stop-time is mm/dd/yy-hh:mm:ss.

The cut contains intervals with outages (status code "1"). For example, if a cut had 10 intervals in succession with outage status codes ("1") starting with the interval ending 01/15/98-6:14:59, then the message would read:

OUTAGES: 10 @ 01/15/98 6:14:59

NONNORMAL: n @ mm/dd/yy-hh:mm:ss
where n = number of consecutive non-normal intervals, beginning with the interval whose recorded stop-time is mm/dd/yy-hh:mm:ss.

The cut contains non-normal intervals (status code "2"-"9"). For example, if a cut had five consecutive intervals with non-normal status codes, starting with the interval ending 05/26/98-12:14:59, then the message would read:

NONNORMAL: 5 @ 05/26/98 12:14:59

HIGH DEMAND: n @ mm/dd/yy-hh:mm:ss
where n = number of consecutive high-demand intervals, beginning with the intervals whose recorded stop-time is mm/dd/yy-hh:mm:ss.

The cut contains intervals with demand greater than that specified in the HIGH Validation Environment File Command. For example, if a cut had four intervals in succession whose demands exceeded the HIGH demand limit, starting with the interval ending 04/02/98-23:59:59, then the message would read:

HIGH DEMAND: 4 @ 04/02/98-23:59:59

LOW DEMAND: n @ mm/dd/yy-hh:mm:ss
where n = number of consecutive low-demand intervals, beginning with the intervals whose recorded stop-time is mm/dd/yy-hh:mm:ss.

The cut contains intervals with demand less than that specified in the LOW Validation Environment File Command. For example, if a cut had five intervals in succession whose demands were less than the LOW demand limit, starting with the interval ending 03/18/98-11:59:59, then the message would read:

LOW DEMAND: 4 @ 03/18/98-11:59:59

(EXTERNAL) TIME UNDERLAP: x INTERVALS
where x = number of intervals in the underlap (gap)

(EXTERNAL) TIME OVERLAP: x INTERVALS
where x = number of intervals in the overlap

Either of these two messages indicates that the cut has failed the **"Recording Period Match-Up Test."** The gap or overlap in minutes between the stop-time of the cut and its following cut exceeded the allowable limit. For example, if a cut stops at 10/07/97-09:53:59 and the following cut starts at 10/07/97-9:29:59, there is an overlap of 15 minutes. If the overlap tolerance is 15.

(EXTERNAL) METER UNDERLAP: x UNITS
where x = size of the underlap (gap)

(EXTERNAL) METER OVERLAP: x UNITS
where x = size of the overlap

Either of these two messages indicates that the cut has failed the **"Meter Reading Match-Up Test."** If the meter stop-reading of the current cut is less than the meter start-reading of the following cut, a gap or underlap exists. If the meter stop-reading of the current cut is greater than the meter start-reading of the following cut, an overlap exists. If the gap or overlap is greater than the allowable limit, the cut fails the test.

(EXTERNAL) UNIT OF MEASURE DISCREPANCY

(EXTERNAL) SECONDS PER INTERVAL DISCREPANCY

Either of these messages indicates that the cut has failed the **"Merge Attribute Match-Up Test."** Before Oracle Utilities Load Analysis will make cuts available for analysis, the unit-of-measure and seconds-per-interval fields for the current cut must be equal to the corresponding fields for the following cut. If either of these fields is not equal, the cut fails.

If such a merge attribute discrepancy is a valid condition, then you can force the current cut to merge by setting the cut's merge flag to "YES" during the editing process.

> **Note:** Oracle Utilities Load Analysis can store up to 10 validation messages for one cut. When you revalidate the cut, the old messages will be overwritten.

Step 2: Determine Type of Edit to make for each error

Once you have diagnosed the problem using the validation reports and messages, you will need to determine a remedy. Your objective is to correct any problems so that all cuts will pass Validation; in other words, so that the Validation Program will mark all cuts both Internally and Externally Valid. (This is indicated by a V in the STATUS IN and EX columns in the Load Data Validation Execution Log, and a YES for the Validation Flags in a Load Data Report.) The exception is the last cut in a cut series, which may be marked Externally Invalid, because external validation testing cannot be performed on that cut.

Often there are many possible solutions to a problem, and how you will fix problem cuts will depend on the policies of your facility. Check with your Load Research Supervisor — there may be recommended remedies for the problems you need to solve.

Step 3: Create the Editor Control File containing your Edit Commands (TGX31A.CTL)

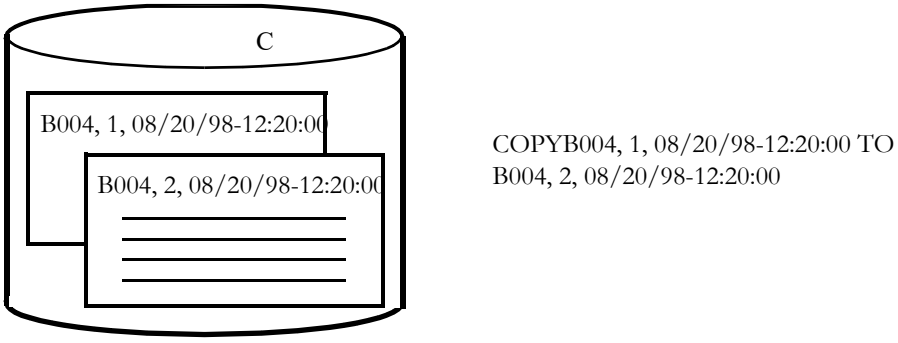
Now you are ready to actually create the edit commands using a special "edit command language." Your commands will make up the Editor Control File.

> **Note:** The Validation Program automatically outputs an "Edit Key File" for cuts that failed validation and require editing. You can use this file as the skeleton for your Control File. A sample of an Edit Key File is shown below.
>
> KEY A1234,2,081198124500
> KEY G1950,1,071398072100
> KEY C1957,3,011698173000

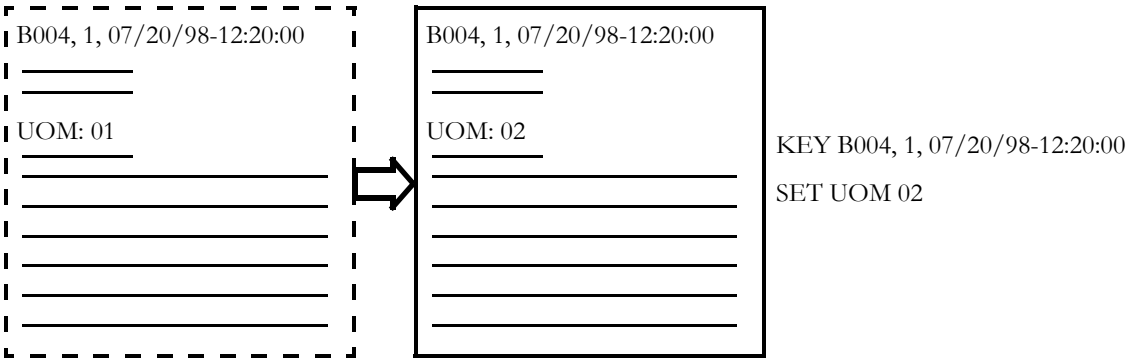
You will use two types of commands to create the Editor Control File (see Figure 9-1):

- **Cut commands** — Use these commands to modify the cut key or select a cut for further processing. Think of the cut commands as a way of manipulating *cuts in the database*.

- **Correction commands** — Use these commands to modify a cut's descriptive fields or interval data. Think of the correction commands as a way of changing *information within a cut*.



Cut commands — think of cut commands as a way of manipulating cuts in the CLDB. For example, the copy



Correction commands — think of correction commands as a way of changing data within a cut. For

Figure 9-1 Editor Cut vs. Correction Commands

You construct the Editor Control File by building "blocks" of commands — each block is a series of commands for a single cut. For each cut, enter cut commands first, then correction commands. You can input up to 25 correction commands in a block. You need only the first three characters to indicate the command title (see Figure 9-1.)

Important Note: You must *always* precede Correction commands with the "KEY" Cut Command.

In other words, you must first identify the cut you want to modify by inputting the KEY Command, then specify the change or changes you want to make using one or more correction commands.

Summary of Edit Commands

Table 1 and Table 2 list all of the edit commands. Each of these commands is described in detail in the section **Editor Command Descriptions** on page 9-12.

Table 1: Cut Commands

Command Format	Application	Example
CHAnge key1 TO key2	Changes one or more components of a cut's key. (i.e., customer-id, channel, and/or start-time) You must specify the full keys.	CHA B004, 1, 07/20/98-12:20:00 TO B004, 2, 07/20/98-12:20:00
COPy key1 TO key2	Creates a duplicate of an existing cut and assigns a new key to the duplicate. key1 = cut to be copied; key2 = new key You must specify the full keys.	COP B004, 1, 08/20/98-12:20:00 TO B004, 2, 09/20/98-12:20:00
EGAp custid, channel [start] [stop]	Scans through a cut series to determine gaps. If detected, gaps can be filled via the NEW correction command and edited using other correction commands.	EGAP N1923, 1 5/01/06 05/31/06 ...
ERAse key	Completely erases a cut from the CLDB, including original copy and edit trails. You must specify the full key. USE THIS WITH CAUTION!!!	ERA B007, 1, 08/20/98-12:20:00
KEY customer-id, channel(s), start-time [,ORIGINAL]	Selects a cut for modification by the Correction Commands. Specify ORIGINAL if you want modifications applied to original version. Any edited versions will be discarded.	KEY B007, 1, 08/20/98-03:45:00, ORIGINAL
NEW custid channel start stop FROM custid channel start [meter start] [meter stop]	Synthesizes a new cut. Subsequent correction commands can be applied to the newly synthesized cut.	NEW N1723 1 5/13/06-13:57:00 6/14/06-12:20:59 FROM N1723 1 4/12/06-13:50:00 14022.5 14700.0
REStore key	Restores a cut to its original, unedited version. The edited version and any edit trails are discarded.	RES B005, 2, 07/16/99-02:30:00
SPLit key AT time [NEWkey customer-id [,channel]]	Splits a cut into two cuts at a time within the data range of the original. If you specify NEWkey, creates two completely new cuts; if not, first portion of split retains original key and replaces existing cut.	SPL B007, 1, 09/20/98-03:30:00 AT 09/30/98-02:59:59 NEW B007,2

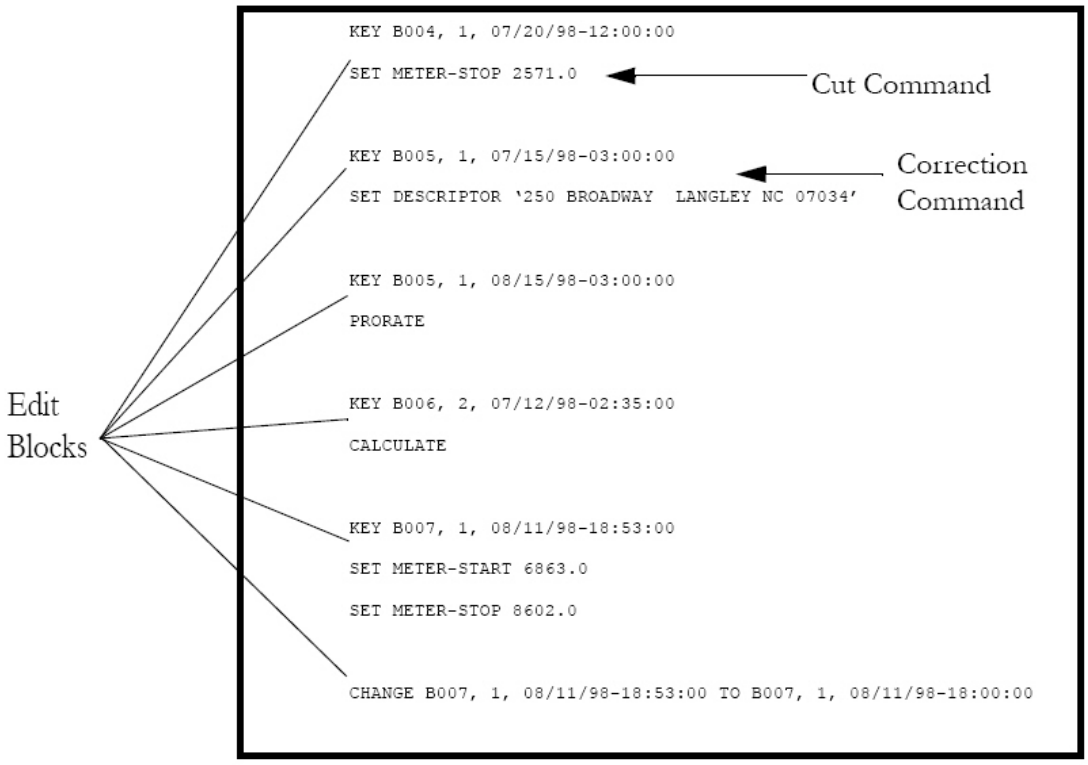


Figure 9-2 Sample Editor Control File

Here is a sample Editor Control File showing the edit command structure. Remember that each edit block pertains to a single cut.

Table 9-2: Correction Commands

Command Format	Application	Example
ADDition {*time1* | **START**}{*time2* | **STOP**} *z*	Adds a constant "z" to each interval in the specified time span.	ADD START 08/20/98-12:20:00 3
AVErage [*start* | **APPend**] [*stop* | *#ints*] [**I** | **O**] [**AW** | **AD**] [**W1** *Start* | -3] [**W2** *Stop* | 3] [**Q** [*q* | 8]] [**S** [*s* | J]]	Inserts a new period of intervals into an existing cut, or overwrites existing intervals. The intervals of data used for inserting or overwriting consist of averages computed over interval data from the cut being edited, or from another time period for the same customer id and channel.	AVE 0 APP 10
CALculate	Calculates the actual stop-time of a cut based on its start-time, number of recorded intervals, and Seconds per interval.	CAL
DELete {*time1* | **START**} {*time2* | **STOP** | **DO** *n*}	Deletes all intervals in a cut that fall within the specified time span.	DEL 08/20/98-12:20:00 STOP
INSert {*time1* | **APPend**} {*time2* | **DO** *n*} {**Value** *z* | **From** *key* **AT** *time3*}	Inserts intervals into a cut. You can either specify a value "z" to be added "n" times, or you can copy and paste a series of intervals from the same or another cut. Use APP to put intervals at the end of a cut.	INS APP D 5 V 3
INTerpolate *time1* {*time2* | **DO** *n*} [**Q** [*q* | 8]] [**S** [*s* | J]]	Performs linear interpolation of intervals from time1 through time2, or for n intervals. Q defines the worst acceptable status code for the intervals before and after the interpolation range. S specifies the status to be assigned to the interpolated intervals.	INT 09/22/98-03:30:00 09/22/98-14:59:59
MODify *time* [**Status** *s*] **Value** *z1* [*z2 z3... z29*]	Changes a sequence of interval values to a new sequence of values "z1"..."z29" beginning at "time." Optionally, can also change status to "s" for affected interval values.	MOD 05/11/99-10:45:00 S 9 V 12 15 13
MULtiply {*time1* | **START**}{*time2* | **STOP**} *z*	Multiplies all interval values that fall within the specified time span by a constant "z."	MUL START 04/13/98-11:00:00 3

Table 9-2: Correction Commands

Command Format	Application	Example
OVErwrite *time1* {*time2* | **DO** *n*}{**Value** *z* | **Status** *s* | **Value** *z* **Status** *s* | **From** *key* **AT** *time3*}	Overwrites intervals in a cut. You can specify a value "z" to overwrite "n" intervals, or you can copy and paste a series of intervals from the same or another cut.*	OVE 05/20/98-09:50:00 D 5 V 2
PROrate STAtus [*s* | *] [**MAX** *x* | 32760] [**MIN** *n* | 0] [**MET** *ngy*]	Prorates one or more data intervals so the cut's interval energy is consistent with its meter energy. Optionally, you can specify that only status codes of a certain value "s" be prorated. You can also specify a peak value not to be exceeded, a minimum value not to be exceeded, and metered energy to override the metered energy in the cut.	PRO J MET 2500.0
REAding *time r* [*#dials* [.*#decimals*]]	Use this command to specify a meter reading at a given time within the cut's range. The Editor will recalculate the cut's meter start and stop readings to conform to this information and the cut's total interval energy.	REA 09/24/98-14:17:00 2437
REMark [*remark*]	Add notes of up to 188 characters to an edit trail. Recommendation — record the initials of the person making the edits.	REM CORRECTED BY JDL
SET *field value*	Manually overrides storage flags or resets descriptive fields other than cut keys, load data, or status codes. See Table 3 later in this chapter, or see the *Quick Reference Guide,* for a complete list of eligible fields.	SET ARCHIVE YES SET UOM 02
SMOoth [**HIGh** | **LOW**] [**Value** *z* | 0] [**DO** *n* | 1] [**Status** *s* | **K**]	Eliminates spikes or dips by linear interpolation between the intervals preceding and following and setting the status code.	SMO LOW V 0
STAtus {*old_sta* | *} *new_sta* **DATe** {*start* [*stop*]} **INT** {*low_int* [TO *high_int*]}	Changes all occurrences in a record of status code *old_sta* to *new_sta*. If *old_sta* is an asterisk (*), all status codes are changed to *new_sta*. The DATe command can be used to limit the range of the function to a specified date range, from *start* to *stop* (or the end of the cut if omitted) The INT command can be used to change only those intervals of a specified valie (*low_int*) or whose value falls within a specified range (*low_int* TO *high_int*)	STA 1 L

Guidelines for Using the Edit Commands

Table 1 and Table 2 list all of the edit commands and summarize their applications. The notes below give you some general guidelines for constructing the commands. (Each of the commands is described individually in detail in the next section.)

1. **Full Key**

   Many of the edit commands call for a "full key." A "Full key" is customer-id, channel, and start-time.

2. **Inputting Times**

   Many of the edit commands require you to identify a time, i.e., a cut's start-time or a data interval's ending time. The two formats indicated below are always acceptable and interchangeable, although to maximize readability and minimize input errors, the first is preferable:

   - mm/dd/yy-hh:mm:ss (for example, 01/23/99-00:00:00)
   - mmddyyhhmmss (for example, 012399000000).

   In the KEY Command it is permissible to use just the start-date, but *only* when there is just one cut or set of cuts on a particular day.

3. **Specifying Interval Data Values for Modification**

   Many of the correction commands enable you to change one or a series of interval data values in a cut. You must identify which values you want to change by specifying parameters in the command. You can do this in a couple of ways:

   - **"Time1" "Time2"** — You can identify the ending time of the first and last interval in the series to be changed, e.g., "time1 time2". You must input the complete date and time.
   - **"START" "STOP"** — If the first interval in the group you want to change is the first interval in the cut, you can input "START" instead of typing out the time. Similarly, if the

last interval in the group you want to change is the last interval in the cut, you can input "STOP".

- **"DO n"** — You can indicate the first interval in the series (using either its ending time or START) and the number of intervals after it you want changed using "Do n", where n = the number of intervals to be changed. You can enter "D" for Do.
- **"Value z"** — Identify the value to be added, etc. to each specified interval by "Value = z", where z is the constant. You can abbreviate value by just using the letter "V".

4. **Patching Interval Data**

   The EGAP, INSERT and OVERWRITE commands enable you to "copy and paste" interval data from one cut to another. You can copy a series of interval data values from the same cut or any other cut in the CLDB. The intervals can even be from two contiguous cuts. You use **"From key AT time"** and **"Do n"** to specify the intervals to be copied, where "key" is the key of the source cut, "time" is the ending time of the first interval to be copied, and n is the number of intervals to be copied.

5. **Special Note about Overriding Storage Flags with the SET Command**

   You can use the SET Command to override Oracle Utilities Load Analysis's Merge and Archive fields. More specifically, you can designate a cut that has not passed validation for transfer to the ALDB by specifying "SET ARCHIVE YES"; or for merging and transfer to the ELDB by specifying "SET MERGE YES". *Use this feature with caution, because there is no safeguard to prevent an internally invalid cut from being marked for merging.*

Editor Command Descriptions

The following section provides a detailed description of each edit command. The commands are listed in alphabetical order within type — Cut Commands appear first, followed by Correction Commands. Remember that *Correction Commands must be preceded by the Cut Command "KEY".*

All editor command keywords may be entered in an abbreviated form. Each editor command may be invoked by entering three or more characters of the command name. The syntax scanner only checks the first three characters to ensure it qualifies as one of the uniquely abbreviated command forms.

> **Note:** When any command that alters interval data values is executed, if the cut contains a non-zero Pulse Multiplier (indicating that it was created using the X110 Direct Input Program), the Pulse Multiplier and Pulse Offset fields are reset to zeros.

CHANGE Command

The CHANGE Command changes one or more components of a cut's key; i.e., the customer-id, channel, and/or start-time. The format of the command is:

CHAnge key1 TO key2

The full key, including the hour, minute, and second of the start-time, must be specified. Multiple channels are not permitted. Correction Commands may not follow the CHANGE Command.

The first time the CHANGE Command is used, the old key ("key1") is stored in a special field that is printed whenever the cut is reported. Repeated use of this command on the same cut, however, will not alter the special field. If the cut was edited before the CHANGE Command was issued, the key of the backup copy of the cut is also changed, and the same reference key is placed in the special field.

If the start-time field is altered, the CHANGE Command *automatically recalculates the cut's stop-time based on the new start-time and actual intervals*. This ensures a consistent date range for validation purposes. If the new key ("key2") is already in use, the command will be aborted.

The following are valid examples of the CHANGE Command:

CHA A001,1,01/15/98-12:35:00 TO B0001,1,01/15/98-12:35:00
CHANGE B1920,1,10/19/98-09:45:00 TO B1920,1,10/29/98-09:45:00
CHA C0002 1 071998091100 TO C002 2 071998091100

COPY Command

The COPY Command synthesizes cuts. It copies the descriptive fields and load data values from an existing cut to a new cut with a different key. The format of the command is:

COPy key1 TO key2

The full key, including the hour, minute, and second of the start-time, must be specified for both keys. Multiple channels are not permitted. The word "TO" is optional. No action will be taken if the new key ("key2") already exists. No Correction commands may follow the COPY Command.

If the start-time of the new key differs, the COPY Command *automatically recalculates the cut's stop-time based on the new start-time and actual number of intervals*. This ensures a consistent date range for validation purposes. If the new key is already in use, the command will be aborted.

The COPY Command also copies all edit trails and archive records.

The following are valid examples of the COPY Command:

COP A0001,1,01/15/98-12:35:00 TO B0002,1,01/09/98-11:32:00
COPY C0002 1 071998091100 TO C0002 2 071998091100

EGAP Command

The EGAP command scans a cut series for gaps between cuts, and creates an empty cut to fill the gap to which subsequent commands can be performed as desired. The format of the command is:

EGAp customer-id, channel [start-time] [stop-time]

Note: The EGAP command by itself would not perform any edits.

where:

customer-id channel - are the recorder ID and the channel number of the cut series to scan for gaps.

[start-time] [stop-time] - are optional start time and stop time values. If specified, the command only scans the cut series within this time range.

All commands that can be used under a KEY command can be used under an EGAP command. The only difference is that for correction commands that require a START and/or STOP time parameter, the keywords "START" and "STOP" must be used.

Sample EGAP command edit:

```
EGAP N1723, 1 05/01/82 05/31/82
  REMARK EXTERNAL GAP EDIT
  AVERAGE START STOP O AD W1 -8 W2 -4
  PRORATE
```

The above example will perform the following:

Synthesize a cut(s) any gap(s) (missing cuts) found for the cut series N1723, 1 between period 05/01/82 and 05/31/82, then perform the following:

1. Perform a REMARK edit on the synthesized cut
2. Perform an AVERAGE edit on the synthesized cut
3. Perform a PRORATE edit on the synthesized cut

ERASE Command

The ERASE Command erases a cut from the database. The format of the command is:

ERAse key

The full key, including the hour, minute, and second of the start-time, must be specified. No Correction Commands may follow the ERASE Command. The ERASE Command completely removes all traces of the cut from the database, including the backup copy and edit trail, if any. The only way to recover an erased cut is by re-inputting it through the Input Function. For this reason, **the ERASE Command should be used with discretion**. If there is any doubt about whether the cut might be used for some other purpose, use the Correction Command SET to designate the cut for archiving.

Examples of the ERASE Command are:

ERA A0001,1,01/15/98-12:35:00
ERASE A0004,2,011598101500

KEY Command

The KEY Command selects a cut for detailed editing. It is usually followed by one or more Correction Commands. The format of the command is:

KEY customer-id, channel(s), start-time [,ORIGINAL]
The key command by itself would perform no edits.

Single or multiple channels for the same customer-id and start-time may be designated for editing. *Multiple channels must be entered within parentheses* and separated by plus signs (+) with no spaces between; e.g., "(1+2+3)". An abbreviated form of the start-time ("mm/dd/yy") can be used if there is only one cut (or one set of cuts with different channels) on the day; otherwise, the full start-time ("mm/dd/yy-hh:mm:ss") should be used.

The first time the KEY Command is issued, a backup copy of the original cut is automatically created. This backup will remain unaffected by the editing process. If the KEY Command is used again, all subsequent edits will be applied to the "active" version.

However, if ORIGINAL is specified, any previous edits applied to the cut will be nullified, and the edit trail will start from scratch. ***Note:*** The same effect can be achieved by using the RESTORE Command, but the latter cannot be followed by Correction Commands.

The following are valid examples of the KEY Command:

KEY A0001,1,05/15/98
KEY A0001,1,05/15/98-14:37:00,ORIGINAL
KEY B1920,(1+2),06/11/98
KEY C0002 1 071998091100

The following are invalid forms of the KEY Command:

KEY A0001,05/15/98 **(No channel)**
KEY A0001,1,5/18/98-14:30:00 **(illegal month)**
KEY B1920,1,2,06/11/98 **(Multiple channels not enclosed within parentheses)**

NEW Command

The NEW command synthesizes a new cut based on an existing cut in the database. The format of the command is:

KEY customer-id channel start-time stop-time FROM customer-id channel start [meter start] [meter stop]

where:

- customer id channel start stop - is the new custid, channel, start time, and stop time of the new cut. Fully qualified start and stop times must be specified.
- customer id channel start - is the key of existing cut in the database. The new cut will use this cut as a template for all basic header information such as SPI, UOM, Descriptor, etc.
- [meter start] [meter stop] - optional meter start and meter stop readings. If supplied, these values will be applied to the new cut's meter start/stop readings.

All subsequent commands that can be used under a KEY command can be used under the NEW command.

RESTORE Command

The RESTORE Command restores an edited cut to its original form from the backup copy. It nullifies all edits and erases the edit trail. The format of the command is:

REStore key

The full key, including the hour, minute, and second of the start-time, must be specified. Multiple channels are not permitted. No Correction commands may follow the RESTORE Command. Examples of the RESTORE Command are:

RES A0001,1,01/15/98-12:35:00
RESTORE A0003,2,04/11/98-21:02:00

SPLIT COMMAND

Like the COPY and NEW Commands, the SPLIT Command synthesizes cuts. It splits a designated cut into two at either a time or at a metered energy that you specify. Meter stop/start readings at the time of the split are calculated based on the total metered energy of the designated cut. The format of the command is:

SPLitkey AT time [NEWkey customer-id [,channel]]

key AT METer value [NEWkey customer-id [,channel]]

The "key" parameter must contain the full key, including hour, minute, and second of the start-time, of the cut to be split. The word "AT" is optional. No Correction commands may follow the SPLIT Command.

When splitting based on a time, the time entered must be between the start and stop times of the designated cut. This time becomes the start time of the second cut created by the split; a time one second before this is calculated as the stop time of the new/changed first cut.

When splitting based on a metered energy reading, the value entered must fall within the specified cut's metered start energy reading ([meter start * meter multiplier] + meter offset) and the cut's total energy.

The optional NEWkey parameter is used to assign to the two cuts resulting from the split a customer-id and/or channel different from those of the original cut. The first parameter following NEW is the new customer-id; a channel may follow. If you specify a new key, the split creates two new cuts in the database; if not, the first cut generated by the split has the same key as the original cut, and so replaces it. If the key of the second cut from the split (or the key of the first when NEWkey is specified) is already in use, the command will be aborted.

The following are valid examples of the SPLIT Command:

```
SPL A0001,1,01/15/98-12:35:00 AT 02/01/98-12:15:00
SPL A0002,1,03/18/98-11:45:00 AT 03/29/98-12:01:00  NEW  A0002,3
SPL A0002,1,03/18/98-11:45:00 03/29/98-12:00:00  NEW  A0004
```

ADDITION Command

The ADDITION Command adds a constant to the data intervals of a cut. The format of the command is:

ADDition {*time1* | **START**}{*time2* | **STOP**} *z*

The intervals affected by the command are bounded by the "time1" and "time2" parameters, inclusive. The keywords "START" and "STOP" denote the first and last intervals, respectively. *If explicit times are provided, the complete date/time of the intervals must be specified.*

> **Note:** This same functionality may be accomplished using the Totalizing Reporter, in the Load Data Transformation programs (X430, Y620).

The parameter "z" can be in any numeric format and *may be negative*.

The status code of each affected interval is automatically set to "L". *If either specified time falls outside the cut's range, the command is aborted.*

The following are valid examples of the ADDITION Command:

```
ADD START STOP 10/* ADD 10 TO INTERVAL VALUES
ADD START 12/03/97-09:15:00 -2/* SUBTRACT 2 FROM INTERVAL VALUES
```

AVERAGE Command

The Average Command allows you to insert a new period of intervals into an existing cut, or to overwrite existing intervals in a cut. The intervals of data used for inserting or overwriting consist of averages computed over interval data from the cut being edited, or from another time period for the same customer-id and channel. Two types of averaging may be performed: averages of intervals occurring on specific days of the week, or averages computed over weekdays or weekend days. The format of the AVErage Command is:

AVErage [*start* | **APPend**] [*stop* | *#ints*] [**I** | **O**] [**AW** | **AD**] [**W1** *Start* | -3] [**W2** *Stop* | 3] [**Q** [*q* | 8]] [**S** [*s* | 1]]

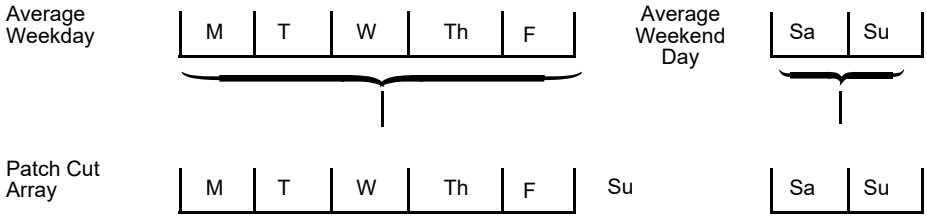
where:

The parameter "I" is used to indicate that intervals are to be inserted, and "O" (the default) indicates that intervals are to be overwritten. The parameters "I" or "O" must follow the "START" and "STOP" parameters (or "APPend" and "#ints").

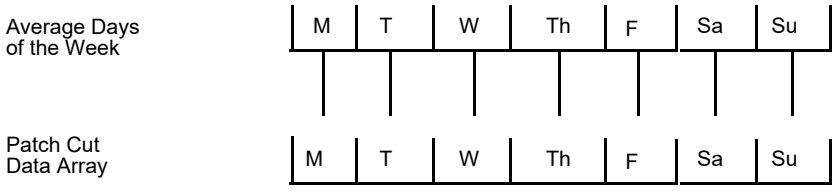
The parameters "START" and "STOP" are used to indicate the date and time boundaries of the intervals to be overwritten, or the position in the cut at which intervals are to be inserted. A complete date and time must be specified for the "START" and "STOP" parameters. In Overwrite mode, you must either specify both a start and stop date and time, or a start date and time and the number of intervals to be overwritten (the "#ints" option is used to specify the

number of intervals). In Insert mode, you may specify either a start date and time plus the number of intervals to insert, or the parameter "APPend" to indicate that the new intervals are to be added to the end of the cut. When the "APPEND" parameter is used, the number of intervals to be inserted must also be given.

The parameters "AW" and "AD" are used to indicate the type of average to be computed for intervals to be inserted or overwritten. "AW" indicates Average Weekday/Weekend, meaning that average weekday and average weekend calculations will be performed over the days in the period identified for averaging, and these averages will be written over corresponding weekdays or weekends in the data period being edited, or inserted into the cut. The following diagram shows how the data will be patched:



"AD" indicates that an Average Day calculation will be performed for each day of the week over the period identified for averaging, and these average days will be written over the corresponding data to be edited or inserted into the cut as follows:



The second "Start" parameter is used to locate data to be averaged within the current cut or from a different time period for the same customer-id and channel. The start of this averaging period is identified by stating the number or weeks before or after the "START" date and time parameter to begin averaging data. The default is '-3' weeks. Any value from -52 to 52 is valid. The second "Stop" parameter identifies the number of full weeks of data to compute the average over. The default is 3 weeks, and any value from 1 to 12 is valid. For example, if the cut with key

N1723, 1, 02/14/98-15:26:00

is being edited, and the user has specified the options

AVErage 03/01/98-10:00:00 03/07/98-10:00:00 O AD W1 -4 W2 2

then Oracle Utilities Load Analysis will overwrite the intervals in the cut from 10:00:00 a.m. on March 1, 1998 to 10:00:00 a.m. on March 7, 1998 with data representing average days calculated over the two weeks of data occurring from February 1, 1998 at 10:00:00 a.m. (4 weeks before March 1, 1998 at 10:00:00 a.m.) to February 14, 1998, 10:00:00 a.m.

If you want to specify an averaging period different from the default (START -3 or STOP 3), then the labels W1 and W2 need to preface the number of weeks specified. For example:

AVE 050398133000 053198240000 O AD W1 -8 W2 4

The "QUALITY" parameter, "Q", may be used to specify a quality code for the worst acceptable data to be included in computing averages. The default is '8', indicating that any data intervals with a status code of '9' will not be included when calculating averages. The "STATUS" parameter, "S",

may be used to specify a status code to be assigned to all inserted or overwritten intervals. Acceptable status codes are J - Z and 0 - 8. The default status code is 'J'.

CALCULATE Command

The CALCULATE Command calculates the actual stop-time of the cut based on the start-time, the number of recorded intervals, and the seconds-per-interval. The re-computed stop-time is stored in the cut's stop-time field. The format of the command is:

CALculate

This command is used to fix an incorrectly transcribed stop-time, or to force the cut's expected intervals to match the recorded intervals.

Certain Cut commands (namely, CHANGE and COPY) may automatically invoke this command.

DELETE Command

The DELETE Command deletes data intervals. The format of the command is:

DELete {*time1* | **START**} {*time2* | **STOP** | **DO** *n*}

Note that the "DO" keyword has a single-letter abbreviation. The intervals affected by the command are bounded by the "time1" and "time2" parameters, inclusive. The keywords "START" and "STOP" denote the first and last intervals, respectively. *If explicit times are provided, the complete date/time of the intervals must be specified.*

The alternate form of the second parameter indicates that "n" intervals should be deleted starting with the interval ending at "time1". The value of "*n*" *must be positive.*

The DELETE Command automatically recalculates the stop-time of the cut based on the start-time, new number of intervals, and seconds-per-interval. If "time1" is outside the range of the edited cut or if "time2" is outside the range of available source cuts, the command is aborted.

The following are valid examples of the DELETE Command:

```
DEL 05/11/98-10:45:00 STOP/* DELETE INTS FROM 5/11 THRU STOP
DELETE START D 128/* DELETE 128 INTERVALS FROM START
```

Oracle Utilities Load Analysis will automatically shift data toward the start-time. To delete intervals from the beginning of a cut and adjust the start-time, use the CHAnge Command to create a new key, then delete the initial intervals.

INSERT Command

The INSERT Command inserts data intervals. There are two formats of this command:

INSert {*time1* | **APPend**} {*time2* | **DO** *n*} {**Value** *z* | **From** *key* **AT** *time3*}

Note that the "DO", "VALUE", and "FROM" keywords have single-letter abbreviations. In the first format, the interval *before* which the data is inserted is specified by the "time1" parameter. The keyword "APPEND" denotes that the new intervals should be placed at the end of the cut. *If an explicit time is provided, the complete date/time of the interval must be specified.*

The "time2" parameter allows for a stop-time to indicate the last interval to be inserted (time1 and time2 identify where intervals are inserted).

The "DO" parameter indicates that "n" intervals should be inserted. The value of "n" must be positive.

The "VALUE" parameter specifies the data value assigned to the inserted intervals. The status code of each inserted interval is set to 'J'. If the status code of the existing interval denoted by "time1" is "1" (power outage), it will also be reset to 'J'.

The "FROM" parameter indicates a data patch from a cut in the CLDB. The "key" identifies the customer-id and channel of a cut or cuts from one of three sources:

- The same cut
- Another cut (including an adjacent channel for the same customer)
- A set of contiguous cuts.

The source is automatically determined by the software based on the indicated key and number of intervals required. The first interval from which the source data is composed is specified by the "time3" parameter. Exactly "n" intervals (indicated by the "DO" parameter or by the time1-time2 range) are inserted. The intervals in the patched section of the edited cut are given a status code of "L".

The source cuts *must be internally valid, unless the patching is performed from the same cut*. In the latter case, the source data is taken from the *active record* that existed prior to entering the current edit block.

The INSERT Command *automatically recalculates the stop-time of the cut based on the start-time, the new number of intervals, and the seconds-per-interval*. If "time1" is outside the range of the edited cut or if "time3" is outside the range of available source cuts, the command is aborted.

The following are valid examples of the first format of the command:

```
INS 05/11/98-10:45:00 DO 7 VALUE 0/* INSERT '0' FOR 7 INTERVALS
INS APP D 96 F A0001,1 AT 061198180000/* APPEND 96 INTERVALS
```

The second format of the INSERT Command can be used after the first format has been specified, and after itself. It causes the preceding INSERT Command to be repeated. The following example causes the first command to be repeated twice:

```
INSERT 05/11/98-10:45:00 D 7 V 0/* INSERT '0' FOR 7 INTERVALS
INS =/* INSERT ANOTHER 7 INTERVALS OF '0'
INSERT =/* AND YET ANOTHER
```

INTERPOLATE Command

The Interpolate Command estimates data in a range of intervals by linear interpolation between the intervals occurring immediately before and after the range (referred to hereafter as the "first point" and "second point," respectively).

The format of the command is:

INTerpolate *time1* {*time2* | **DO** *n*} [**Q** [*q* | 8]] [**S** [*s* | J]]

The required "time1" parameter indicates the time of the first interval to be estimated and must be within the time period of the cut being edited. The complete date/time must be specified.

The "Do" parameter is used to specify the number of intervals whose data is to be replaced; the value "n" must be at least 1 and may not exceed the number of intervals from "time1" to the end of the cut. Alternatively, you may specify the "time2" parameter to indicate the end of the interpolation range; "time2" may not be before "time1" or after the stop time of the cut. The interpolation range may not constitute the cut's entire time range, i.e., you may not interpolate the whole cut with this command.

The optional "Q" parameter establishes the quality test: the worst acceptable status code for the intervals before and after the interpolation range (first point and second point); the default is '8'.

The optional "S" parameter is used to specify the status code to be assigned to the interpolated intervals; the default is 'J'.

If the interpolation range is in the middle of the cut (i.e., includes neither its first nor its last interval), the interval immediately preceding the interpolation range is used as the first point from which to interpolate, and the interval immediately following the interpolation range is used as the second point. If either of these intervals fails the quality test, the interpolation is not performed.

If the interpolation range begins with the first interval of the cut, the program will attempt to use the last interval of the preceding cut in the series, if any, as the first point from which to interpolate. If the preceding cut is not present, or if its last interval fails the quality test, the value of the second point is used for the first point, resulting in all interpolated intervals receiving this same value (flat load). However, if the second point fails the quality test, the interpolation is not performed.

If the interpolation range ends with the last interval of the cut, the value of the first point is used for the second point, resulting in all interpolated intervals receiving this same value (flat load). However, if the first point fails the quality test, the interpolation is not performed.

Linear interpolation is performed using the following formulas:

1 point $x_{(n+1)} = x_{(n)} + (x_{(n+2)} - x_{(n)} / 2$

2 points $x_{(n+1)} = x_{(n)} + (x_{(n+3)} - x_{(n)} / 3$

3 points $x_{(n+2)} = x_{(n)} + (x_{(n+3)} - x_{(n)} / 3$

MODIFY Command

The MODIFY Command modifies a series of data intervals and, if specified, their status codes. The format of the command is:

MODify *time* [**Status** *s*] **Value** *z1* [*z2 z3... z29*]

Note that the "TIME" parameter is *required* and specifies the interval ending time at which to start modifying data with the values provided. The "STATUS" and "VALUE" keywords have single-letter abbreviations. The intervals affected by the MODIFY Command begin with "time" and continue for as many intervals as there are values supplied following the "VALUE" keyword. At least one value must be supplied.

The "STATUS" parameter is optional and specifies the status to be assigned to all of the values being entered. Status codes must be in range from 'J' to 'Z' or '0' to '9'.

The "VALUE" parameter is required and specifies the data values to modify successive intervals beginning with "time". *Up to 29 data values can be entered, if they can fit onto one 200-character line.* The values specified cannot extend beyond the STOP TIME of the cut.

The following are valid examples of the command:

```
MOD 05/11/98-10:44:59 STATUS P VALUE 12 15 13 24 19 3
MODIFY 04/12/98-07:59:59 V 0 0 0 125 0 0 0 200
```

MULTIPLY Command

This same functionality may be accomplished using the Totalizing Reporter, in the Load Data Transformation programs (X430, Y620).

The MULTIPLY Command multiplies data intervals of a cut by a constant. The format of the command is:

MULtiply {*time1* | **START**} {*time2* | **STOP**} *n*

The intervals affected by the command are bounded by the "time1" and "time2" parameters, inclusive. The keywords "START" and "STOP" denote the first and last intervals, respectively. If explicit times are provided, the complete date/time of the intervals must be specified.

The parameter "*n*" can be in any numeric format. The resulting value of the multiplied data values will be *rounded to an integer value*.

The status code of each affected interval is automatically set to 'L'. If either time falls outside the cut's range, the command is aborted.

The following are valid examples of the MULTIPLY Command:

```
MUL START STOP 10/* MULTIPLY BY 10
MUL START 12/03/98-09:15:00 1.414/* MULTIPLY BY 1.414
MULTIPLY START STOP 5.1/* MULTIPLY BY 5.1.
```

OVERWRITE Command

The OVERWRITE Command overwrites data intervals and/or status codes. The format of the command is:

OVErwrite *time1* {*time2* | **DO** *n*} {**Value** *z* | **Status** *s* | **Value** *z* **Status** *s* | **From** *key* **AT** *time3*}

> Note that the "DO", "VALUE", "STATUS", and "FROM" keywords have single-letter abbreviations. The intervals affected by the OVErwrite Command are bounded by the "time1" and "time2" parameters, inclusive. If explicit times are provided, the complete date/time must be specified.

The alternate form of the second parameter indicates that "n" intervals should be overwritten starting with the interval ending at "time1". The value of "*n*" *must be positive*.

The "VALUE" parameter specifies the interval value assigned to the overwritten intervals. The "STATUS" parameter specifies the status code assigned to the overwritten intervals. The code must be a *single character equal to or below 'J'* in the Oracle Utilities Load Analysis status code sequence (*i.e., a status code worse than "J"*; see Figure 3-2). Either "VALUE" or "STATUS" or both may be specified. If "VALUE" is not specified, the data values of the overwritten intervals are not changed. If "STATUS" is not specified, the status codes of the overwritten intervals are set to 'L'.

The "FROM" parameter indicates a data patch from a cut in the CLDB. The "key" indicates the customer-id and channel of a cut or cuts from one of three sources:

- The same cut
- Another cut (including an adjacent channel for the same customer)
- A set of contiguous cuts.

The source is automatically determined by the software based on the indicated key and number of intervals required. The first interval from which the source data is copied is specified by the "time3" parameter. Exactly "n" intervals (indicated by the "DO" parameter or derived from "time1" and "time2") are overwritten and assigned the status code 'L'.

The source cuts *must be internally valid, unless patching is performed from the same cut*. In the latter case, the source data is taken from the active record that existed before entering the current edit block.

If any time is outside the range of the edited or available source cuts, the command is aborted.

The following are valid examples of the OVErwrite Command:

```
OVE 05/11/98-10:45:00 05/12/98-10:44:59 VALUE 0 STATUS P
OVERWRITE 04/12/98-00:00:00 D 96 F A0001,1 AT 06/11/98-18:00:00
```

PRORATE Command

The Prorate Command is used to adjust one or more data intervals so the cut's interval energy is consistent with its metered energy. The original Prorate Command adjusted all intervals in the cut, or optionally allowed you to specify that only intervals with status code "s" be adjusted. The enhanced Prorate Command provides three additional parameters, which are optional:

- Peak value not to be exceeded
- Minimum value not to be exceeded
- Metered energy to override the metered energy in the cut.

The format of the enhanced Prorate Command is:

PROrate STAtus [*s* | *] [**MAX** *x* | 32760] [**MIN** *n* | 0] [**MET** *ngy*]

where:

The parameter "s" is an optional status code. If "s" is specified, only those intervals with status codes identical to "s" are prorated; if "*" is specified (or no status code is specified), all the intervals are prorated.

The "MAX" parameter is used to specify a maximum value not to be exceeded during proration. Any data intervals with values larger than "x" will not be prorated, and no prorated interval will have a value larger than "x" as a result of being prorated. Similarly, the "MIN" parameter is used to specify a minimum value not to be exceeded during proration. No data intervals with values less than "n" will be prorated, and no prorated interval will have a value less than "n" after being prorated. Both the "MAX" and "MIN" parameters are optional.

The "MET" parameter is used to specify an energy value that will override the metered energy when calculating a prorate constant for each interval to be prorated. The prorate constant is calculated as follows:

$$Prorate\ Constant = Energy\ Value \times \frac{(Meter \text{ Energy - Interval Energy}}{Total \text{ Energy of Prorated Interval}}$$

READING Command

With the Reading Command you specify a meter reading and the time at which it was taken. Based on this information, the meter multiplier and offset, and the total interval energy of the cut, the Editor calculates the appropriate Meter Start and Stop Readings for the cut.

The format of the command is:

REAding *time reading* [#*dials* [.#*decimals*]]

where:

The "time" parameter indicates the time of the meter reading, which must be within the time range of the cut being edited.

The required "reading" parameter indicates the meter reading, and must be numeric in format.

The program makes use of the optional "#dials" and "#decimals" parameters only if, in conjunction with the cut's energy and other attributes (see above), the "time" and "reading" parameters entered imply a Meter Start Reading that is negative. The "#dials" parameter indicates the total number of dials (digits) on the meter for which the reading is entered, and must be a positive integer no greater than 7. The "#decimals" parameter indicates the number of decimal places represented on the meter. If entered, "#decimals" must be a non-negative integer no greater than "#dials". If "#dials" is entered without "#decimals", the latter is assumed to be 0. If "#dials" and/or "#decimals" are entered and #dials - #decimals is sufficiently large, the Editor attempts to replace a derived negative Meter Start Reading with a recalculated Meter Start Reading

representing no more than one rollover of the meter described. If this is not possible, the command is aborted.

The following are valid examples of the REAding Command:

```
REA  05/22/98-09:14:59  2534
REA  05/22/98-09:14:59  25436  5
REA  05/22/98-09:14:59  85049.23  7.2
```

REMARK Command

The REMark Command places remarks in an edit trail. The format of the command is:

REMark [*remark*]

A remark is *limited to 188 characters* and may be left blank. It is ignored for execution purposes.

The following are valid examples of the REMark Command:

```
REM SOURCE DATA OBTAINED FORM ADJACENT CHANNEL
REMARK
REM CORRECTED ON JAN 3, 1999
```

SET Command

The SET Command resets various descriptive fields in a cut other than the customer-id, channel, start-time, stop-time, load data, or status codes. The format of the command is:

SET *field value*

The complete list of fields and permissible values appears in Table 3. Hyphens or underscores may be used in the field names. Abbreviations that can be used for certain fields are given in parentheses.

In addition to resetting descriptive fields, the SET Command is also used for manually overriding storage flags. A cut that has not passed validation may be designated for transfer to the Archive Load Database (ALDB) by specifying:

```
SET  ARCHIVE  YES
```

Alternatively, a cut that has not passed validation may be designated for merging and transferred to the Archive Load Databases (ALDB) by specifying:

```
SET  MERGE  YES
```

This feature of the SET Command should be used with caution. There is no safeguard to prevent an internally invalid cut from being marked for merging. **Setting the Merge flag to "YES" also sets the Archive flag to "YES", and setting the Archive flag to "YES" also sets the Merge flag to "YES". For the reverse operations, setting the archive flag to "NO" resets the merge flag, but setting the merge flag to "NO" does not reset the archive flag.** Note that the actual transfer of the cut to the Archive Load Database does not take place until the Scan, Archive/Delete programs are run.

Some examples of the SET Command are:

```
SET MER YES
SET DESCRIPTOR ORACLE UTILITIES
SET ARCHIVE YES
```

Table 9-3: Set Command Summary

FIELD NAME	PERMISSIBLE VALUE
UOM	3 numeric characters (see *Appendix A: Oracle Utilities Unit of Measure Codes*)
DESCRIPTOR (DES)	1-72 characters
DESCRIPTOR1 (DES1)[a]	1-40 characters
DESCRIPTOR2 (DES2)[b]	1-40 characters
SECONDS-PER-INTERVAL (SPI)	Integer (60, 300, 900, 1800, 3600, 86400)
METER-MULTIPLIER (METER-MULT)	Any positive numeric format
METER-OFFSET	Any numeric format
METER-START	Non-negative numeric format
METER-STOP	Non-negative numeric format
PULSE-MULTIPLIER (PULSE-MULT)	Any non-negative numeric format, or NULL
PULSE-OFFSET	Any numeric format
Note that the Pulse Multiplier and Pulse Offset fields are valid only for editing cuts that were initially created with these fields present.	
TIME-ZONE-STANDARD-NAME (or TZSN or TZS)	1-32 characters. Must be one of EST, CST, MST, or PST.
ARCHIVE	YES ON OR NO OFF
MERGE	YES ON OR NO OFF
POPULATION	YES ON OR NO OFF
WEIGHT	Non-negative integer
Note that the Population and Weight fields are valid only for editing statistic cuts.	Any weight between 0 and 1

a. If you specify only the SET DESCRIPTOR1 (SET DES1), SET DESCRIPTOR2 (SET DES2) is set to blanks.
b. SET DESCRIPTOR2 (SET DES2) *must* be preceded by a SET DES1 Command.

SMOOTH Command

The Smooth Command provides you with a technique for eliminating spikes (abnormally high values) or dips (abnormally low values) in the data profile that exceed expected limits. Occurrences of spikes or dips in the data are expected to be of short duration. As a result of smoothing, bad values (spikes and dips) are replaced in the data by linear interpolation between the intervals occurring before and after the bad values.

The format of the command is:

SMOoth [**HIGh** | **LOW**] [**Value** *z* | 0] [**DO** *n* | 1] [**Status** *s* | **K**]

where:

The "HIGH" or "LOW" parameter indicates that high or low values are to be smoothed. Only one option can be used per edit, and the default is LOW.

The "VALUE" parameter is used to specify the upper (abnormally high) or lower (abnormally low) bound used to determine which data values will be smoothed. The default value is 0.

The "DO" parameter is used to specify how many high or low values occurring in sequence are to be smoothed. The value "n" must be greater than or equal to 1 and cannot exceed the number of intervals in the cut being edited. The default value is 1.

The optional "STATUS" parameter is used to specify a status code that will be assigned to all smoothed data intervals. The default status code is 'K'.

When data smoothing is performed, the cut's interval data is searched from beginning to end. If a sequence of values exceeds the maximum value specified by the user, and sequence length (the DO parameter) is less than or equal to the number of intervals in the cut, then if:

SET field value

- The first interval of the cut exceeds the maximum, set the first value to the specified maximum value and linearly interpolate the "bad" values to the first good value after the sequence
- The last interval of the cut exceeds the maximum, set the last value to the specified maximum value and linearly interpolate the "bad" values from the last good value before the sequence
- A data sequence in the middle of the cut exceeds the maximum, linearly interpolate the "bad" values between the last good value before and after the sequence.

Smoothing of low values is performed in the same way as high values, except that the data is checked for sequences of interval data with values below the stated minimum.

Linear interpolation is performed using the following formulas:

1 point $x_{(n+1)} = x_{(n)} + (x_{(n+2)} - x_{(n)})/2$

2 points $x_{(n+1)} = x_{(n)} + (x_{(n+3)} - x_{(n)})/3$

$x_{(n+2)} = x_{(n)} + 2(x_{(n+3)} - x_{(n)})/3$

etc.

STATUS Command

The STAtus Command resets all occurrences in a cut of a given status code to another status code designated in the command. The format of the command is:

STAtus {*old_sta* | *}* *new_sta* **DATe** {start [stop]}
INT {low_int [TO high_int]}

The status to be changed may be any character; if there are no such status codes in the cut, no changes will be made. The *new_sta* can be any valid status code or 'BLAnk'. Note that a status code of '9' may be assigned only to an interval with a data value of 0 (zero); thus if the *new_sta* is 9 and any interval containing the status-to-be-changed has a non-zero value, the command will pass the syntax check but will not be executed. If you wish to change *all* status codes in a cut to the *new_sta*, code an asterisk (*) as the status-to-be-changed. Status codes can be

Use the DATe command to limit the function to operate over a specified date range. The <*start*> parameter is required if DATe has been supplied. The <*stop*> parameter is optional and designates when the program stops the search. If omitted, the program searches/replaces to the end of the cut. The <*start*> and <*stop*> do not have to be within the range of the cut. If this is the case the program will not error and will do nothing, however it will print out a warning message.

Use the INT to only change the status code if the associated intervals are of a specific value or within a range of interval values. Use <*low_int*> to set a single interval value or the lowest interval value that will trigger the status code change and <*hig_int*> to designate the upper limit of the interval range. These two values are inclusive. If specifying a single interval value, omit the TO keyword and terminate the command.

If a blank status code is required for either old or new status code, the user must enter a ' ' or the three letters BLA. New and old status codes may be specified using optional single quotes.

Some examples of the STAtus command include the following:

Example 1: The user wishes to change blank status codes to status code 8 over 10/05/17 to 12/01/18 for a cut specified previously using the KEY command, when intervals are 0 - 10

```
STA BLA 8 DAT 10/05/18 12/01/18 INT 0 TO 10
```

Example 2: The user would like to change all status codes 8 to Q in a cut, but only if the interval values are greater than 0 and less than 100.

```
STA 8 Q INT 0 TO 100
```

Example 3: The user would like to change all status codes 8 to Q within the cut but only where the interval value is 0.

```
STA 8 Q INT 0
```

The following is a sample Editor Control File illustrating various load data edit commands. The edit command blocks have been delimited by blank lines and commented by the Remark Command (REM).

The Editor Environment File determines the mode of processing. Create this file with two commands:

```
KEY Z1212,(1,2,3),01/01/98-12:00:00
REM CORRECT FAULTY STOP TIME ALL 3 CHANNELS.
CAL

KEY Z1234,1,01/04/98
REM OVERWRITE INCORRECT DATA WITH DATA FROM
REM THE NEXT WEEK AND PRORATE.
OVE 01/05/98-12:00:00 01/06/98-11:59:59 FROM Z1234,1 AT 01/12/98-12:00:00
PRO L

KEY A1234,1,12/12/97
REM CORRECT AN OUTAGE.
INS 12/14/97-11:00:00 DO 4 VALUE 0

KEY D1212,1,01/02/98
REM PRORATE ENTIRE CUT.
PRO

CHA A1234,2,01/01/98-12:00:00 TO A1234,1,01/02/98-12:00:00

KEY A1235,4,01/01/98-01:00:00
REM RESET INCORRECT CONSTANTS.
SET METER-MULT 1.1
SET METER-OFFSET 100
```

Figure 9-5 Another Sample Editor Control File

Step 4: Create the Editor Environment File (TGX31B.ENV)

It is recommended to use the default selection when using the editor. The commands for the editor environment file are as follows:

EXEcute [OFF | ON]

AUDit [OFF | ON]

PRInt [ECOnomize | FULl]

MERge [YES | NO]

GRA

- **EXECUTE** — This feature checks your edits without applying them to the database. In the EXECUTE OFF mode, the Editor simply scans the edit commands in the Control File for errors. No commands are executed. In the EXECUTE ON mode, the correctly constructed commands are executed and the incorrectly constructed commands are rejected. EXECUTE ON is the default.

  It is a good idea to first scan your Control File using the OFF mode, correct any errors, then execute the commands by rerunning the program using the ON mode. This can also be done automatically by the Load Data Editor Syntax Scan Program (X320).

- **AUDIT** — This command determines whether or not the Editor maintains an edit trail and backup. In the AUDIT ON mode, the Editor automatically keeps a copy of the original cut and an edit trail containing the date of each edit and the command itself. In the OFF mode, the original is discarded and there is no edit trail — only the edited version is saved.

  The OFF mode is useful for certain storage operations that will not alter the descriptive fields or load data — such as selecting a cut for archive. It is also recommended that you turn the AUDIT OFF when you are making the same edit to a large percentage of the cuts in the CLDB. For example, if your translator miscalculated the stop-time for a large number of cuts, you would edit each cut with the identical command. Rather than wasting space in the database storing the same edit command with every affected cut, you should turn the AUDIT OFF and possibly make a note of the change in just one record.

  Otherwise, it is recommended that, when editing the CLDB, you always use AUDIT ON (default). When editing the ELDB, use AUDIT OFF (default).

- **PRINT** — This command allows for the reduction of the amount of paper used for the Load Data Editor Syntax and Execution Logs. In the Economize mode the dotted and blank separator lines between cuts is omitted. In the full mode the program prints the Load Data Editor Syntax Log and Execution Log with full separation.

- **MERGE** — An optional command. When set to "YES", the program will merge cuts when using the AVErage, INSert, NEW, or OVErwrite commands regardless of validation status. When set to "NO" (default), the program will reject invalid cuts and return an error message indicating the cuts are not merge-able.

- **GRA** — This command determines that a graphics file is produced when running this program.

Step 5: Run the Load Data Editor Procedure (X310 or X320)

Once you have created all of the necessary inputs — the Environment File and the Control File — you are ready to run the Load Data Editor Procedure. Use X310 or X320 if you just want to check the syntax of your editor commands. The procedure X310 will run the Load Data Editor, Load Data Validation, and Load Data Reporter programs.

Editor Processing

Edit commands are grouped into edit blocks, each of which details an editing operation to be performed on a cut. The Load Data Editor reads each block of edit commands and processes it in two phases. The first phase scans the block for syntax errors. If a syntax error is detected, appropriate diagnostic messages are printed on the Load Data Editor Syntax Log and the processing of the block is terminated. Only the first phase is performed in the EXECUTE OFF mode.

The "EXECUTE ON" mode proceeds to the second phase: if the block is free of syntax errors, it is passed to the execution phase that performs the actual editing of the cut. If an error is detected during the execution phase, appropriate diagnostic messages are printed on the Load Data Editor Execution Log. The CLDB is updated only if the block is executed successfully.

Cuts are initially stored in the CLDB as active records. In general, when the Editor is directed to edit a cut, it makes a copy of the active record in the CLDB and performs the edits on that copy. If the editing operation is successful, the edited copy replaces the active record in the CLDB. If errors are encountered during editing, the copy is discarded and the active record remains unchanged in the CLDB.

The Editor stores the edits successfully performed on a cut in a CLDB record called an "edit trail." The edit trail associated with an edited cut is displayed whenever the cut is reported by the Load Data Reporter.

If a cut is being edited for the first time, the Editor makes an additional copy of the unedited active record and stores it in the CLDB as the cut's inactive record of original data. *It also blanks out the cut's validation message area, removing all previous validation messages.* The inactive record of original data is a backup of the cut's unedited data, and can be accessed by the Editor to restore that data.

Thus, every edited cut has a copy of its original data and a list of all the edits made to that data.

Step 6: Identify Remaining Errors and Return to Step 2 if Necessary

If the reports show remaining invalid cuts, return to Step 2 and repeat the process.

Sample Editing Session

To help introduce you to the editing process, we will go through a sample editing session, step-by-step. In this example, it will be our job to correct the invalid cuts input to the CLDB in *Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs*. We will go through the same process described on the previous pages, but we will use specific examples to illustrate each step. You may also elect to use the Interval Data Manager to perform edits.

Step 1: Review Validation Reports for Invalid Data

At the end of an input procedure, Oracle Utilities Load Analysis automatically produces a series of reports that identify invalid cuts. We will look at some of those now.

Step 1A: Look at the Load Data Validation Program Execution Log

This report gives us a good overview of the work ahead. Six cuts require editing — five are internally invalid and one is externally invalid. (The cuts with a line in the External Status column are at this point considered externally invalid because there is no following cut to check against — but they do not require editing.) You may also create a Key Generator query to create a list of cuts that you wish to edit.

Step 2: Look at the Load Data Reports and Determine Edits

For each invalid cut, Oracle Utilities Load Analysis automatically produces a detailed "Load Data Report." (See examples next page.) We will look at each one carefully to identify the problem, and determine the most appropriate edit.

- **B004, 1, 07/20/98-12:20:00** — The validation messages tell us that the Energy Discrepancy Test and the Non-Normal Intervals Test have detected problems in the cut. The ratio of metered energy to interval energy is 0.974, and the cut has one non-normal interval at 08/13/98-20:59:59. When we check these values against the validation criteria established in the Load Data Validation Environment File, we see that one non-normal interval is acceptable per cut — so the cut actually passes the Non-Normal Intervals Test. *However*, the ratio of metered energy to interval energy is below the acceptable range, indicating a problem. We examine the meter start and stop readings, and observe that numbers in the stop reading have

been entered incorrectly. If we change its value from 2517.0 to 2571.0, the meter and interval energy will both equal 2060. In other words:

(stop-reading — start-reading) x meter multiplier + meter offset = meter energy

(2571.0 - 511.0) x 1.0 + 0.0 x 744 = 2060.0

We can change the stop-reading using the SET Command. The edit block to correct this cut will be:

KEY B004, 1, 07/20/98-12:20:00
SET METER-STOP 2571.0

(We will input this edit in the next step.)

- **B005, 1, 07/15/98-03:00:00** — We notice that this cut was not assigned a descriptor, and we want to add the customer's address. We can do this using the SET Command. The edit block would be:

```
KEY B005, 1, 07/15/98-03:00:00
SET DESCRIPTOR 250 BROADWAY LANGLEY NC 07034
```

  We should also correct the outages identified in the validation messages, but we will intentionally overlook this problem to show you how Oracle Utilities Load Analysis will catch the mistake.

- **B005, 1, 08/15/98-03:00:00** — Here is another cut that failed the Energy Discrepancy Test. In this case, we decide that the difference between the meter and interval energy is small enough to distribute equally over all of the intervals in the cut. Oracle Utilities Load Analysis's PRORATE Command will do this for us automatically. The edit block would be:

  KEY B005, 1, 08/15/98-03:00:00
  PRORATE

- **B006, 2, 07/12/98-02:35:00** — The validation message tells us that this cut failed the Number of Intervals Test. Based on the recorded stop-time, the cut should have had 720 intervals, but 744 were recorded (the cut is 24 intervals longer than expected). We decide that the recorded stop-time is incorrect and that we want to change it to the computed stop-time. The CALCULATE Command will do this for us automatically. The edit block would be:

  KEY B006, 2, 07/12/98-02:35:00
  CALCULATE

- **B007, 1, 07/11/98-18:53:00** — The overlap problem in this cut is caused by an external validation error. By looking at the next cut in the series we can see that it is missing meter readings. We will edit the meter readings in the 08/11/98 cut to correct the problem. We will also change the 8/11/98 cut start time to fix the time underlap (gap) shown on this cut.

- **B007, 1, 08/11/98-18:53:00** — The meter data for this cut is missing, so we obtain the correct values from the Metering Department. We can use the SET Command to correct the record. The edit block will be:

  KEY B007, 1, 08/11/98-18:53:00

  SET METER-START 6863.0

  SET METER-STOP 8602.0

The start time is changed to correct the time underlap (or gap) between the two cuts in the series. The command to modify the start time is:

CHANGE B007, 1, 08/11/98-18:53:00 TO B007, 1, 08/11/98-18:00:00

Step 3: Create Editor Control File (TGX31A.CTL)

Now that we have determined our edits, we will put them together in the Editor Control File. This file will be one of the two inputs for the Load Data Editor Program. For each cut we want to modify, we will enter an edit command block consisting of a Key Command and (optionally) one or more Correction commands.

> **Note:** We have intentionally included a mistake in this file to show you what happens when Oracle Utilities Load Analysis encounters a "bad" edit command. ("DESCRIPTOR" in the second edit block is misspelled.))

```
   KEY B004, 1, 07/20/98-12:20:00

   SET METER-STOP 2571.0

   KEY B005, 1, 07/15/98-03:00:00

   SET DECRIPTOR 250 BROADWAY  LANGLEY NC 07034

   KEY B005, 1, 08/15/98-03:00:00

   PRORATE

   KEY B006, 2, 07/12/98-02:35:00

   CALCULATE

   KEY B007, 1, 08/11/98-18:53:00

   SET METER-START 6863.0

   SET METER-STOP 8602.0

   CHANGE B007, 1, 08/11/98-18:53:00 TO B007, 1, 08/11/98-
18:00:00
```

Figure 9-6 Editor Control File

Step 4: Create the Editor Environment File (TGX31B.ENV)

The Environment File is the second input for the Load Data Editor Program. We will leave the defaults in place. The EXECUTE ON tells Oracle Utilities Load Analysis to apply our edits to the cuts (rather than scanning them first), and the AUDIT ON tells the system to save the original version as an inactive record along with an edit trail.

Step 5: Run the Load Data Editor Procedure

Submit the job using X310. Oracle Utilities Load Analysis will not only apply the edits to the cuts, but will also revalidate and report them.

Step 6: Check Output Reports

Because the Editor Procedure automatically runs the Editor, Validation, and Reporting programs, it produces many reports. We will look at each one to ensure that the programs executed properly, and that our edits were successfully applied to the cuts.

```
                    EXECUTE                 ON

                    AUDIT                   ON
```

```
PRINT                    FULL
```

Figure 9-7 Editor Environment File

- **Load Data Editor Environment Report** — Confirms that the Editor Program was executed under the desired conditions.
- **Load Data Editor Syntax Log** — Lists back our edits and points out any mistakes we might have made when constructing the edit commands. The report tells us that the fourth line was incorrect — we misspelled "DESCRIPTOR". We will have to correct the error in the next iteration of the procedure.
- **Load Data Editor Execution Log** — Indicates that all other edits were successfully applied to the cuts.
- **Load Data Validation Program Environment Report** — Confirms the criteria Oracle Utilities Load Analysis used for the Validation Tests.
- **Load Data Validation Program Execution Log** — Lists back the edited cuts that underwent re-validation (along with any other cuts in the CLDB from the same cuts series), and the results of the validation tests.
- **Load Data Validation Program Run Summary** — Reiterates that one cut is still internally invalid. Eight are externally invalid, but looking back at the Load Data Validation Program Execution Log, we see that it is because they have no following cuts. This means that Oracle Utilities Load Analysis was unable to perform External Validation tests, and the cuts do not require additional editing at this point.
- **Load Data Reporter Environment Report** — Shows the Environmental File parameters in effect when Oracle Utilities Load Analysis produced the following Load Data reports. (You will learn what these parameters mean and how to change them in *Chapter 12: Using the Time Series (X400) and Load Data (X410 and X420) Reporters*.)
- **Load Data Reporter Execution Log** — Summarizes the reporting options in effect for each reported cut series. (Again, you will learn how to change these parameters in *Chapter 12: Using the Time Series (X400) and Load Data (X410 and X420) Reporters*. Occasionally, you will need additional information to diagnose the invalid cuts, and changing these parameters can give you that information.)
- **Load Data Reporter Summary Report** — Gives us a quick overview of the Load Data Reporter Run. The report total refers to the number of *cut series*.
- **Load Data Reports** — Detailed reports on any cuts that failed validation a second time, and any cuts in the same cut series. This time, Oracle Utilities Load Analysis produced two Load Data Reports:
  - **B005, 1, 07/15/98-03:00:00** — The report for this cut is unchanged from the first time we saw it. Oracle Utilities Load Analysis produced the report a second time because we overlooked a problem that required correction — the cut failed the Number of Power Outages Test. We will have to correct this problem in the second iteration of the procedure.
  - **B005, 1, 08/15/98-03:00:00** — This report shows what happens when edits are successfully applied to a cut. Notice that the edit flag has changed, there are no validation messages telling us that there is a problem, and the edit trail recorded our changes. The edited version is stored as an "active" record. The original version is saved as an "inactive" record.

Step 7: Repeat the Procedure for Any Remaining Invalid Cuts

Because we have one remaining invalid cut and one edit was faulty, we will have to repeat the process. Remember, creating a database of reliable information is critical to a successful load research program. Return to Step 2.

Chapter 10

Re-Testing Data Quality Using the Cut Series Validation Program (X210, X220)

As you learned in *Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs*, Oracle Utilities Load Analysis's Input Procedures automatically apply the Validation Program to test interval data for quality and completeness as the data is entered into the CLDB. There may be times when you want to run the Validation Program as a separate job to retest data that already resides in the CLDB. For example, your utility might change its standards for acceptable-quality data, in which case you would have to re-validate all data in the CLDB using the new tolerances.

This chapter explains how to run Validation as a separate job. In addition, it provides a more detailed description of the Validation Tests, and of the Automatic Editor feature. This information applies whether you are running Validation as a separate job or as part of another procedure.

The following topics are covered in this chapter:

- **What Does the Validation Program Do?**
- **The Invalid Series Validation Program (X220)**
- **A Closer Look At the Validation Tests**
- **Steps for Using the Validation Program (X210, X220)**
- **The Automatic Editor**
- **Energy Discrepancy Rules**

What Does the Validation Program Do?

The Cut Series Validation Program, X210, checks newly-entered, newly-edited, or user-specified cuts in the Current Load Database (CLDB) to ensure that the data is accurate and error-free. The Cut Series Validation Program checks all data against tolerance parameters specified by your utility in the Validation Environment File (see *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*). Data that fails these standards cannot be transferred to the Archive Load Database (ALDB) without first being edited.

The program performs two categories of validation tests: internal validation and external validation. Internal validation examines the status codes, the start and stop-times, the data values, and the meter readings associated with each cut. External validation examines the match-up of a cut to the preceding cut and to the following cut in its cut series. Because external validation requires a following cut, a cut cannot be externally valid until the data for the next recording period has been entered into the CLDB. Thus, the most recently entered cut in each cut series will always be marked externally invalid.

In order to examine the "fit" between individual cuts in a cut series, the Cut Series Validation Program always examines an entire cut series. *There can be up to 10,000 cuts in a series*. If validation is being performed as part of an Input or Editing procedure, the Load Data Input Program, or the Load Data Editor automatically passes a file of keys identifying the cut series with newly-entered or newly-edited cuts. If you are performing validation as a separate job, you will have to provide the list of keys to be validated.

The Cut Series Validation Program uses two flag fields on each cut in the CLDB: an internal validation flag and an external validation flag. These flags are set to "NO" when a cut is entered into the CLDB by a Load Data Input Program or when a cut is modified by the Load Data Editor. When a cut passes the internal validation tests and/or the external validation tests, the Cut Series Validation Program sets the appropriate flag(s) to "YES". A cut is considered to be "valid" and ready to be transferred into the Archive Load Database only when both of these flags are set to "YES".

A validation message area also exists on each cut's CLDB record. Up to ten validation messages issued for a cut can be stored in this area. When a cut is re-validated, the messages in this area are overwritten. The messages are automatically printed when a cut is reported by the Load Data Reporter.

The Cut Series Validation Program does not report the detailed validation messages that it writes to the CLDB, but instead produces a Control File of cut keys (the Editor Key List FIle) for the Load Data Reporter. The Load Data Reporter reports the status of the validation flags and the contents of the validation message area for each cut series containing an internally or externally invalid cut or, optionally for the invalid cuts only. The CLDB Summary Reporting program, X440, and the ALDB Summary Reporting program, X460, also report the status of the validation flags for cuts in the respective databases. (These reporting programs are described in later chapters in this manual.)

> The Scan, Archive/Delete Procedure, X910 (also described later in this manual) relies on the cut validation flags to determine which CLDB cuts are ready to be transferred to the ALDB. A cut retention parameter tells the Scan Program how many valid cuts must be left in the CLDB for each cut series. All valid cuts beyond this limit are automatically archived in the ALDB, and deleted from the CLDB.

The Invalid Series Validation Program (X220)

The Invalid Series Validation Program is a special case of the Cut Series Validation Program. In this case, the Key Generator program (TGY110) is run as a preliminary step to generate a list of keys of cuts that are internally and/or externally invalid. This key list then serves as the Control

File for the Cut Series Validation Program. Thus, only cut series that contain invalid cuts are subjected to validation.

A Closer Look At the Validation Tests

The following paragraphs describe each of the tests that the Validation Program applies to the data. The tests use the tolerance parameters specified by your utility in the Validation Environment File (see *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*). The program applies two sets of tests: *internal validation* and *external validation*.

Internal Validation

Internal validation checks the internal consistency and quality of a cut. The internal validation flag on each cut's CLDB record indicates whether the cut has passed internal validation. The following internal validation tests are performed on each cut:

- *Number of Intervals Test* — The actual number of data intervals must correspond to the expected number of intervals.
- *Energy Discrepancy Test* — The difference between the meter energy and the interval energy must not be greater than the meter multiplier times the meter multiplier factor,

  and

  The ratio of the meter energy to the interval energy must be within the tolerance range specified by the ENERGY parameter.
- *Uncorrected Power Outages Test* — The number of uncorrected power outage status codes must not be greater than the tolerance level specified by the OUTAGE parameter in the Validation Environment File.
- *Non-Normal Intervals Test* — The number of non-normal intervals must not be greater than the tolerance level specified by the NONNORMAL parameter.
- *Zeros Intervals Test* — The number of consecutive intervals with a value of 0 must not be greater than the tolerance level specified by the ZERO parameter in the Validation Environment File.
- *Spike Intervals Test* — Interval values must not exceed the average of a specified number of peaks in a cut by a percentage greater than or equal to the limits specified by the SPIKE parameter in the Validation Environment File.
- *Dip Intervals Test* — Interval values must not exceed the rolling average of a specified number of preceding intervals by a percentage greater than or equal to the limits specified by the DIP parameter in the Validation Environment File.
- *High and Low Interval Demand Tests* — Interval demands must not exceed limits specified by the HIGH and LOW parameters in the Validation Environment File.

If a cut fails one or more of these tests, it is flagged internally invalid. Each of these tests is explained below.

Number of Intervals Test

The actual number of data intervals in a cut must correspond to the difference between the cut's start-time and stop-time. The Validation Program calculates how many intervals should be present in the cut by subtracting the cut's start-time from its stop-time and converting the difference to interval equivalents. This "expected" value is compared to the number of intervals actually present in the cut. If the expected number of intervals is not equal to the recorded number of intervals, the

cut fails the test, and the following message showing the cut's computed stop-time is written to the cut's validation message area:

(INTERNAL) COMPUTED STOP TIME: mm/dd/yy-hh:mm:ss

As an example of this test, if the cut has a start-time of 01/10/99-14:05:00 and a stop-time of 02/11/99-14:37:59, the time difference is 31 days, 32 minutes, and 59 seconds. If the cut were recorded in 900-second intervals, the time difference corresponds to 2979 intervals. If 2976 intervals were actually recorded, then the cut is either too short or one of the times is incorrect; in either case, the cut would fail this test.

Energy Discrepancy Test

The cut's energy usage based on its meter readings and its energy usage based on the interval data values must be consistent. The cut's meter energy is calculated by the following formula:

$$\text{Meter Energy} = \left[\begin{pmatrix} \text{Meter Stop Reading} \\ - \\ \text{Meter Start Reading} \end{pmatrix} \times \text{Meter Multiplier} \right] + \text{Meter Offset}$$

The cut's total interval energy is calculated by the following formula:

$$\text{Interval Energy} = \sum (\text{Interval Data Value})$$

The cut must pass one of the two energy consistency checks described below.

In the first check, the absolute difference between the two energy values is compared against the meter's resolution (determined by multiplying the meter multiplier by the MULTIPLIER parameter of the Validation Environment File). If the following condition is true, the cut passes the Energy Discrepancy Tests:

$$(\text{Meter Energy} - \text{Interval Energy}) \leq (\text{Meter Multiplier} \times \text{Meter Multiplier Factor})$$

If the first check fails, the ratio of the cut's meter energy to its interval energy is compared to the tolerance range defined by the ENERGY parameter in the Validation Environment File. If the following condition is true, the cut passes the Energy Discrepancy Test:

$$\text{Energy Lower Bound} \leq \frac{\text{Meter Energy}}{\text{Interval Energy}} \leq \text{Energy Upper Bound}$$

If the interval energy is zero, the ratio is undefined and the check fails.

If a cut fails both of these checks, it fails the Energy Discrepancy Test, and the following message is written to the cut's validation message area:

(INTERNAL) ENERGY DIFFERENCE (M - I): d	RATIO (M/I): r)
where **d** is the absolute difference between the cut's meter energy and its interval energy, and **r** is the ratio of the meter energy to interval energy.	

For example, suppose a cut has a net meter reading of 540, a meter multiplier of 2, a meter offset of 0, and 720 intervals. This yields a meter energy value of 1080. If the energy value is 1079, the difference between the meter and interval energy values is 1, which is less than the meter multiplier. If a meter multiplier factor of 1 has been specified, the energy ratio 1080/1079x1 is well within any practical tolerance. Thus, the cut passes the Energy Discrepancy Test on both counts, although actual testing would stop with the first test.

> The Energy Discrepancy Test can be bypassed by using the "ENERGY OFF" Environment File Command.

Uncorrected Power Outages Test

The number of intervals with status codes of "1" must not exceed the limit specified in the OUTAGE parameter of the Validation Environment File. If the number of uncorrected power outages is greater than the limit specified by the OUTAGE parameters, the cut fails this test. A contiguous series of intervals with outage status codes form a "run." For every run of power outage status codes, the following message may be written to the cut's validation message area:

OUTAGES: k AT mm/dd/yy hh:mm:ss
where **k** is the number of intervals in a single run beginning at the specified time.

For example, if a cut had 10 intervals in succession with outage status codes ("1") starting with the interval ending 01/15/98 06:14:59, then the message would read:

OUTAGES: 10 AT 01/15/98 06:14:59

Non-Normal Intervals Test

The number of intervals with non-normal status codes must not exceed the limit specified in the NONNORMAL parameter of the Validation Environment File. For this particular test, a "non-normal" status code is defined as a value from "2" to "9" inclusive. If the number of non-normal status codes is greater than the limit specified by the NONNORMAL parameters, the cut fails this test. A contiguous series of intervals with non-normal status codes form a "run". For every run of non-normal status codes, the following message may be written to the cut's validation message area:

NONNORMAL: n AT mm/dd/yy hh:mm:ss
where **n** is the number of non-normal intervals in a single run beginning at the specified time.

For example, if a cut has five intervals in succession with non-normal codes starting with the interval ending 05/26/98 12:14:59, then the message would read:

NONNORMAL: 5 AT 05/26/98 12:14:59

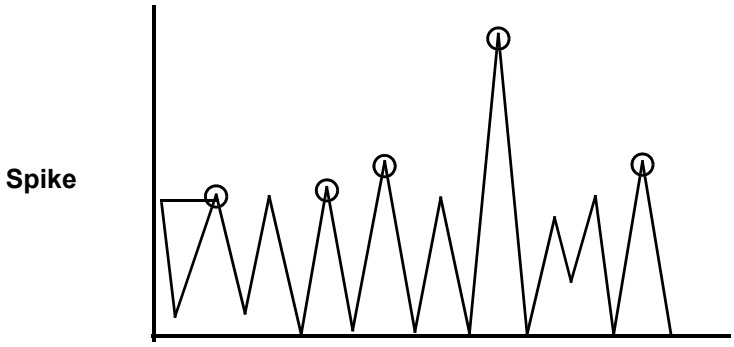
Spike Intervals Test

The cut must not include any abnormally high peaks or 'spikes' based on user-specified limits. Both the number of peaks to be averaged within a cut and a percentage not to be exceeded are defined in the Spike Command in the Validation Environment File. To test a cut's interval data for spikes, the number of peak values specified in the SPIke Command is averaged. The following formula is then used to determine the presence of a spike:

$$\frac{\text{Interval Value} - \text{Average of Peaks}}{\text{Average of Peaks}} \times 100\% \geq \text{Cutoff Percentage}$$

If a cut fails this test, the following message is written into the cut's validation message area:

SPIKE: n @ mm/dd/yy-hh:mm:ss
where **n**=number of consecutive spike intervals in a single run beginning at the specified time.



This figure is a graphical representation of how spike processing works. In the figure, the circled points represent peaks in a cut. Notice that one peak is much higher than the others (an **outlier**). The Spike test will flag this interval as an error.

Dip Intervals Test

The cut must not include any abnormally low values (sudden drops in value) or 'dips' based on user-specified limits. Both the number of intervals over which to compute a rolling average and a percentage not to be exceeded are defined in the Dip Command in the Validation Environment File. To test a cut's interval data for dips, the specified number of intervals immediately preceding the interval being tested are first used to compute a rolling average. The following formulas are then used to determine the presence of a dip:

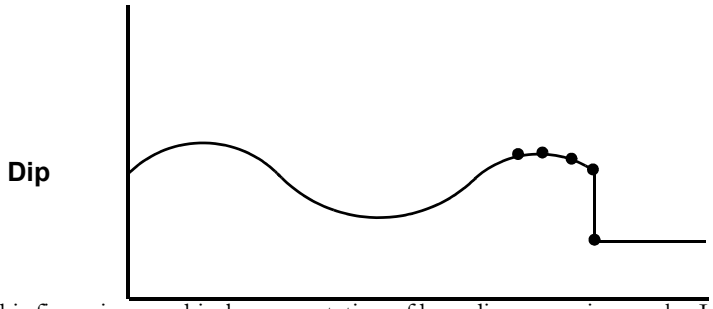
$$\frac{\text{Rolling Average} - \text{Interval Value}}{\text{Rolling Average}} \times 100\% \geq \text{Cutoff Percentage}$$

and

$$\text{Interval Value} < \text{Rolling Average}$$

If a cut fails this test, the following message is written into the cut's validation message area:

DIP: n @ mm/dd/yy hh:mm:ss
where **n**=number of consecutive dip intervals in a single run beginning at the specified time.



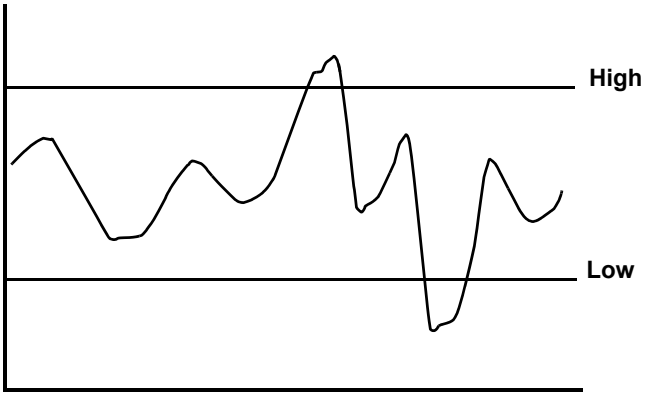
This figure is a graphical representation of how dip processing works. In the figure, notice that the load profile drops suddenly on the right side of the graph. This drop is a 'dip' that will be detected by the dip test and flagged as an error. Such dips can occur many different times within a cut. Note that the Dip test is not appropriate for end-use data, such as appliances that turn on and off.

High Interval Demand Test

The number of intervals with an abnormally high demand value must not exceed the allowed number. Both the value for "abnormally high" and the number of intervals allowed to have a demand greater than that value are defined in the HIGH parameter in the Validation Environment File.

If a cut fails this test, the following message is written to the cut's validation message area:

HIGH DEMAND: n @ mm/dd/yy hh:mm:ss
where **n**=number of consecutive high-demand intervals, beginning with the interval whose recorded stop-time is mm/dd/yy hh:mm:ss.



This figure is a graphical representation of how the High and Low Interval Demand tests work. The lines labeled 'High' and 'Low' represent the user-specified cutoffs for high and low demands. The intervals represented by areas of the load profile that fall outside these boundaries will fail the High/Low Interval Demand tests.

Low Intervals Demand Test

The number of intervals with an abnormally low demand value must not exceed the allowed number. Both the value for "abnormally low" and the number of intervals allowed to have a demand less than that value are defined in the LOW parameter in the Validation Environment File.

If a cut fails this test, the following message is written to the cut's validation message area:

LOW DEMAND: n @ mm/dd/yy hh:mm:ss
where **n**=number of consecutive low-demand intervals, beginning with the interval whose recorded stop-time is mm/dd/yy hh:mm:ss.

Zeros Intervals Test

The number of intervals in a cut whose values are 0 may not exceed a specified limit. If a cut fails this test, the following message is written to the cut's validation message area:

Zeros: n @ mm/dd/yy hh:mm:ss
where **n**=number of intervals with a 0 value, beginning with the interval whose recorded stop-time is mm/dd/yy hh:mm:ss.

External Validation

External validation checks the match-up of a cut to the following cut in its cut series. Because external validation requires the following cut, a cut cannot be externally valid until the data for the next recording period has been entered into the CLDB. The external validation flag on each cut's CLDB record indicates whether or not the cut has passed external validation. The following external validation tests are performed on each cut:

- *Recording Period Match-Up Test* — The stop-time of the current cut must be within the tolerance (specified by the TIME parameter) of the start-time of the following cut.
- *Meter Reading Match-Up Test* — The meter-stop-reading of the current cut must be within the tolerance (specified by the METER parameter) of the meter-start-reading of the following cut.
- *Merge Attribute Match-Up Test* — The merge attributes (i.e., UOM and SPI) of the current cut and the following cut must be the same.

If the cut fails one or more of these tests, it is flagged externally invalid. Each of these tests is explained below.

Recording Period Match-Up Test

The stop-time of the current cut and the start-time of the following cut must be within the underlap (gap) and overlap tolerances specified by the TIME parameter in the Validation Environment File. If the stop-time of the current cut is less than the start-time of the following cut, an underlap or gap exists. If the length of that gap in seconds is greater than the underlap limit specified in the TIME parameter, the cut fails this test. If a gap exists, the following message will be written to the current cut's validation message area:

(EXTERNAL) TIME UNDERLAP: x INTERVALS
where **x** is the number of intervals in the underlap (gap).

If the stop-time of the current cut is greater than the start-time of the following cut, and they do not fall within the same interval, an overlap exists. If the length of that overlap in seconds is greater than the overlap limit specified in the TIME parameter, the cut fails this test, and the following message will be written to the current cut's validation message area:

(EXTERNAL) TIME OVERLAP: x INTERVALS
where **x** is the number of intervals in the overlap.

For example, if a cut stops at 10/07/97 09:44:59 and the following cut starts at 10/07/97 09:29:59, there is an overlap of 900 seconds (15 minutes). If the overlap tolerance is 900 seconds, the cut still passes this test.

Meter Reading Match-Up Test

If meter readings exist for the cut series, the meter-stop-reading of the current cut and the meter-start-reading of the following cut must be within the underlap (gap) and overlap tolerances specified by the METER parameter of the Validation Environment File. If the meter-stop-reading of the current cut is less than the meter-start-reading of the following cut, an underlap or gap exists. If the size of that gap is greater than the underlap limit specified on the METER parameter, the cut fails this test, and the following message will be written to the current cut's validation message area:

(EXTERNAL) METER UNDERLAP: x UNITS
where **x** is the size of the underlap (gap).

If the meter-stop-reading of the current cut is greater than the meter-start-reading of the following cut, an overlap exists. If the size of that overlap is greater than the overlap limit specified in the METER parameter, the cut fails this test, and the following message will be written to the cut's validation message area:

(EXTERNAL) METER OVERLAP: x UNITS
where **x** is the size of the overlap.

For example, if the meter-stop-reading of a cut is 4270 and the meter-start-reading of the following cut is 4280, there is an underlap (gap) of 10 meter reading units. If the underlap tolerance is, say, 1 unit, the cut fails this test.

Merge Attribute Match-Up Test

To enable automatic merging of the recording period boundaries by the ALDB/CLDB Load Data Extraction Program, Y240 (see *Chapter 5* in the *Oracle Utilities Load Analysis Load Data Analysis User's Guide*), the unit-of-measure and seconds-per-interval fields for the current cut must be equal to the corresponding fields for the following cut. If either of these fields is not equal, the cut fails this test, and one or more of the following messages is written to the cut's validation message area:

(EXTERNAL) UNIT-OF-MEASURE DISCREPANCY
(EXTERNAL) SECONDS-PER-INTERVAL DISCREPANCY: n1 FOLLOWED BY n2

If such a merge attribute discrepancy is a valid condition, it is necessary to force the current cut to merge by setting the cut's merge flag to "YES" using the Load Data Editor.

As an example of this test, if the seconds-per-interval field is 900 for the current cut and 1800 for the following cut, the cut would fail this test, and the following message would be issued:

(EXTERNAL) SECONDS-PER-INTERVAL DISCREPANCY: 900 FOLLOWED BY 1800

- **EXEmpt**— Use this command to exempt cuts with specified units-of-measure from undergoing specified Validation tests. This command can be repeated multiple times in a single environment file. Each instance of this command allows the user to specify a Validation test and a list of UOM codes that are to be exempted from that test, leaving the test applicable for cuts of any other UOM code. Multiple EXEmpt commands may be specified for any test if all the UOMs (separated by either commas or blanks) that are to be exempted from it will not fit on a single command. The UOM codes exempted from each test will be shown in the Environment Report on a line or lines immediately following the description of the test. Additionally, a new Environment Report page ("Interpretation Of Exempted Unit Of Measure Codes") will be produced showing, in numeric order, all the numeric UOM codes that have been exempted from any Validation test, together with descriptive names (KWH, KVARH, etc.) for those that are documented Oracle Utilities Load Analysis UOM codes.

  The EXEmpt command uses the following test keywords:

  MET-UNDER—Exemption from the Meter Underlap element of the Meter Reading Match-Up test.

  MET-OVER—Exemption from the Meter Overlap element of the Meter Reading Match-Up test

  ENErgy—Exemption from the Energy Discrepancy test (comparing metered to interval energy)

  TIM-UNDER (or **TIME-UNDER**)—Exemption from the Time Underlap element of the Recording Period Match-Up test

TIM-OVER or (**TIME-OVER**)—Exemption from the Time Overlap element of the Recording Period Match-Up test

OUTage—Exemption from the Uncorrected Power Outages test

NONnormal—Exemption from the Non-normal Intervals test

HIGh—Exemption from the High Interval Demand test

LOW—Exemption from the Low Interval Demand test

SPIke—Exemption from the Spike Intervals test

DIP—Exemption from the Dip Intervals test

ZERo—Exemption from the Maximum Consecutive Zero Intervals test

Steps for Using the Validation Program (X210, X220)

Here is a brief summary of the steps you will follow to run the Validation Program as a separate job. Each of these steps is described in detail on the following pages.

Summary — Using the Validation Program

1. Create the Validation Control File (TGX21A) — a list of keys for the cut series you want to validate.

   In the Invalid Series Validation Program (X220), the Validation Control File is generated by the preliminary Key Generator step. The Key Generator uses its own Control File for this; an example is in Figure 10-1.

2. If necessary, update the Validation Environment File (TGX21B) tolerances for the validation tests.
3. Specify the output file.
4. Run the Validation Procedure (X210, X220).
5. Check output.

Figure 10-1 summarizes the job's inputs and outputs.

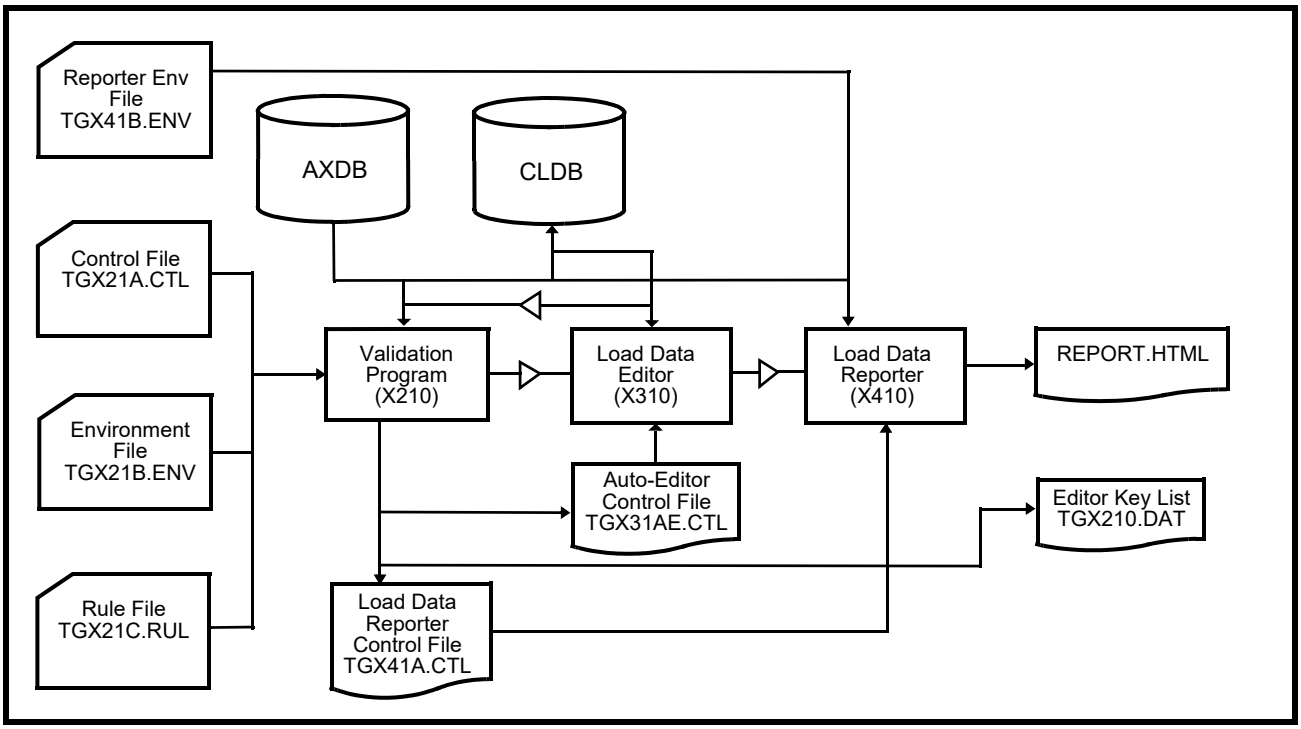


Figure 10-1 Overview of Validation Program

Step 1: Create the Validation Control File (TGX21A)

The Validation Control File contains keys specifying the cut series to be validated. When the Cut Series Validation Program is used as part of the Input or Editor procedures, this file is automatically created by the Input or Editor Program and passed to the Validation Program. In this instance, you must create the file yourself. You can either type it manually or use one of the Key Generator programs described later in this manual. In either case, the format of the file must conform to the specifications described below.

The file must contain one cut series key per line. Each key has the format:

```
customer-id,  channel-number
```

You may separate the two elements (customer-id and channel-number) with blank spaces and/or a comma. You do not need to supply the keys in any particular order — the procedure includes a sort program that automatically sorts them for you. The program automatically discards any duplicate keys.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Step 2: Update the Validation Environment File (TGX21B.ENV) if necessary

The Validation Environment File specifies the tolerances for the validation tests. It determines the quality of data that your utility will use for analysis and, in some cases, billing purposes. *This file should not be changed except under special circumstances and in accordance with the policies of your utility.*

For a complete description of the parameters that make up this file, turn to *How to Create the Validation Environment Files (TGX21B.ENV)* on page 5-2.

If you plan to run with the default file, the file *must* be named TGX21B.ENV and *must* reside in the COMMON\DATA directory.

Step 2A: Update the Edit Rules File (TGX21C.RUL) if used and if necessary

The Edit Rules File, a feature of the Automatic Editor, specifies the edit rules to be applied to certain conditions that may be detected in the cuts being validated, causing Load Data Editor commands to be generated for the correction of those conditions. Like the Validation Environment File, *this file should not be changed except under special circumstances and in accordance with the policies of your utility.*

The Edit Rules File is used only when Automatic Editor processing is to be done. For a complete description of the Automatic Editor, see the section on this subject at the end of this chapter.

The default file should be named TGX21C.RUL. All Rules files *must* reside in the COMMON\DATA directory.

Steps 3 & 4: Specify the Output File and Run the Validation Procedure (X210, X220)

Once you have created or updated the necessary inputs, you are ready to run the program. Use X210.

Validation Processing

The Cut Series Validation Program examines an entire CLDB cut series *(up to 10,000 cuts in the series)*. The Load Data Input programs and the Load Data Editor generate a file of keys for the cut series to be validated. If multiple cuts in a series are entered or edited, redundant cut series keys may be written to this file for validation. To prevent validation of redundant keys that may exist in the Control File, the Validation Program is normally preceded by an automatic sort of the Control File; redundant cut series keys are discarded automatically.

Validation processing proceeds in ascending time order for each cut series. Each cut in the series is read and tested for internal validation. Next, each cut is compared to the following cut for external validation. Finally, all the cuts are rewritten to the CLDB with appropriate values stored in the internal validation flags, the external validation flags, and the validation message areas.

The Load Data Reporter is invoked immediately following Validation, and reports all cuts and/or cut series that were validated and contain invalid data. If there are no invalid cuts, the Load Data Reporter Control File is empty, and 'No valid requests' will be printed on the report.

Step 5: Check Output

The Validation Procedure produces these outputs:

- **Data stored in the CLDB —** The Validation Procedure modifies the CLDB record for each cut that it validated. If a cut passed internal validation and/or external validation, the Validation Program sets the appropriate flag(s) to "YES". Any validation messages issued for a cut during validation are written to its validation message area (and reported in the Load Data Reports produced at the end of the program run). Under the following circumstances, validation messages may be associated with a cut even though it passed the corresponding validation test:
    - Outage intervals existing, although their number is within the OUTAGE parameter's tolerance limit.
    - Non-normal intervals existing, although their number is within the NONNORMAL parameter's tolerance limit.

A maximum of ten validation messages can be stored with a cut. If more than ten messages are generated, the later ones will not be reported.

The **Validation Messages**, which explain what test(s) a cut failed and why, are stored with the cut and printed on the Load Data Reports produced at the end of the Validation Procedure run. A detailed explanation of what these messages mean is provided under **Load Data Reports and Validation Messages** on page 9-5.

- **Execution reports** — The *Validation Environment Report* lists the Validation Environment parameters. The *Validation Edit Rules Report*, if present, lists any Automatic Editor Edit Rules that are in force for this run (see the section on the Automatic Editor at the end of this chapter). The *Validation Execution Log* lists the individual cuts validated within each cut series, and the internal and external validation status of each cut. If a cut is found to be internally invalid, a flag is printed indicating each test that was failed. A *Run Summary* follows the Execution Log, and provides a summary of the number of cuts that were processed, how many were found valid and invalid, and the number of errors and cuts not found. *Note: Samples of these reports can be found at the end of Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs, and under Load Data Reports and Validation Messages* on page 9-5.
- **Files containing the keys of the cuts that failed validation** — The Validation Program produces two key files: A Reporter Control File (TGX41A.CTL) and an Edit Key File (TGX210.DAT).
  - *The Reporter Control File* contains the keys of the cut series that were validated and found to be internally or externally invalid. Any cut that is externally invalid only because the following cut has not been entered into the CLDB is not included in this file. The procedure automatically passes this file to the Load Data Reporter, which produces a report on each key in the file.
  - *The Edit Key File* — The Edit Key File contains Load Data Editor Key commands for the individual cuts that were validated and found to be internally or externally invalid. Any cut that is externally invalid only because the following cut has not been entered into the CLDB is not included in this file. This file may be used as the skeleton for an Edit Command File. A sample of the contents of this file is shown below:

    KEY A1234,2,081198124500
    KEY G1950,1,071398072100
    KEY C1957,3,011698173000
- **File containing Editor commands (conditional)** — This file (TGX31AE) is produced only if an Edit Rules File has been input to the Validation Program, and if Editor commands have been generated as a result of these rules. For further details, see the Automatic Editor section of this chapter.
- **Load Data Reports for all invalid cut series** — Samples of the Load Data Reports can be found at the end of *Chapter 6: Entering and Validating Data Using the Production Input, Direct Input, or Load Data Input Programs,* and at the beginning and end of *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320).*

The Automatic Editor

The Automatic Editor combines the Validation Program and the Load Data Editor into an integrated system that acts upon user-specified rules to correct problem data in Oracle Utilities Load Analysis cuts. When you provide an Edit Rules File (TGX21C.RUL) to the Validation Program, that program uses the rules supplied to diagnose whether certain problems exist in a cut being validated, determines whether the cut is eligible for correction, and generates appropriate Load Data Editor commands to effect the correction. If any such commands are generated, the Validation Program causes the Load Data Editor to be invoked, and the Load Data Editor

Program performs the actual editing. The edited cuts are then submitted to the Validation Program a second time for re-validation.

If you do not provide an Edit Rules File to the Validation Program, no attempt is made to perform automatic editing.

No automatic editing is done when running the Invalid Series Validation Program (X220).

The Edit Rules File (TGX21C.RUL)

The problems that may be addressed by the Automatic Editor are outage periods (intervals with status code '1'), or any other statuses defined as non-normal in the Validation Environment file (TGX21B.ENV), and two types of energy discrepancy. Each of these conditions has its own particular edit rule. This file must be located in the COMMON/DATA folder. You may set up multiple Rule Files; each must have the ".RUL" extension.

Outage and Non-Normal Rules

{**OUTage** | **NONnormal**} {1 | 2 | * } **MAX** *n* { **AVErage** [**AW** | **AD**] [**W1** *start*] [**W2** *stop*] [**Q** *q*] [**S** *s*] | **INTerpolate** [**Q** *q*] [**S** *s*] | **STATus** *s* | **EGAP**}

The OUTage rule is applicable to the outage periods in a cut: sets of consecutive intervals with status '1'. The NONnormal rule is applicable to non-normal periods: sets of consecutive intervals all having the same non-normal status code ('2' through '9'). A specified number (up to 50) or all instances of either type of period in a given cut is subject to automatic editing. The OUTage and NONnormal rules have the same options, which are described below.

Eligibility parameters: {1 | 2 | *n* | * }

The first parameter after the OUT or NON command names must be an integer between 1 and 50, or an asterisk (*). If you enter a 1, the first outage/non-normal period detected in a cut will be eligible for correction under this rule. If you enter a 2, the second such period will be eligible for correction under this rule. If you enter *n*, the *n*th such period will be eligible for correction under this rule. If you enter an asterisk (*), an unlimited number of outage/non-normal periods are eligible for correction under this rule.

MAX n

This is required and specifies the maximum number of intervals that an outage/non-normal period may contain to be eligible for correction under this rule; the parameter "n" must be a positive integer.

Action Parameters: You use the remaining parameters to prescribe the corrective action to be taken for an eligible outage/non-normal period. There are three possible actions: Averaging, Interpolation, and Status assignment. You do not enter parameters specifying the location of the intervals to be edited; the Automatic Editor determines these by examining the cut's data with respect to the Edit Rule's eligibility parameters.

AVErage

This parameter indicates that the data in the intervals of the eligible outage/non-normal period is to be replaced by averaged data from the cut series. The parameters following AVErage correspond to those for the Load Data Editor AVERAGE Command, and are described in *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320)*. They are all optional. If you omit any of them from the Edit Rule, it will be omitted from any Editor AVERAGE Command generated as a result of this rule, and the Load Data Editor will therefore use its default value, as described in *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320)*, for the omitted command parameter.

EGAp

The EGAp parameter indicates the beginning of an EGAP automatic edit block. The series correction commands that follow this parameter have the same syntax as the X310 EGAP series correction commands. For more information about the EGAP command, see **EGAP Command** on page 9-13.

INTerpolate

This parameter indicates that linear interpolation is to be performed between the interval preceding the eligible outage/non-normal period and the interval following it. The parameters following INTerpolate correspond to those for the Load Data Editor INTERPOLATE Command, and are described in *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320)*. They are all optional. If you omit any of them from the Edit Rule, it will be omitted from any Editor INTERPOLATE Command generated as a result of this rule, and the Load Data Editor will therefore use its default value, as described in *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320)*, for the omitted command parameter.

STAtus s

This specifies a status code to be assigned to each of the intervals in the outage/non-normal period. (This is accomplished with an Editor OVERWRITE Command.) The parameter "s" must be a single character in the range J - Z or 0 - 8.

Spike Interpolation Rule

SPIke MAX *n* **[ZERoes]** **[PREceding]** **INTerpolate** **STATUS** *status-code*
[*status-code*] **[SURrounding]**
[SUCeeding]

The SPIke interpolation rule uses the spike intervals test to identify a spike, and then checks for outage intervals that precede it. If an outage interval is found, the spike interval will be interpolated over the preceding zero values for the outage intervals.

SPIke

Required. This is the command to define the rule.

MAX *n*

Required. This command specifies the maximum number of preceding outages prior to the spike for the rule to engage. A numeric positive non-zero value must be specified less than or equal to 10000.

ZERoes / status-code

Optional. Determine the type of qualifying intervals to look for at the spike interval. Specify ZERoes to indicate testing for intervals with zero values. Supply a status-code (single character or single quoted) to look for intervals based on a specific status. Default is ZERoes.

PREceding / SURrounding / SUCeeding

Optional. Determine the direction from which to look for zero intervals / status codes. Specify PREceding to only look for qualifying intervals before the spike. SUCceeding looks for qualifying intervals after the spike. SURrounding will look for intervals before and after the spike. Default is PREceding.

INTerpolate

Required. This command instructs the rule to interpolate over the preceding zero values.

STATUS *<status-code>*

Optional. STATUS is a keyword recognized by the command to set intervals that have been interpolate to a specified status code. The following single character status code must be acceptable to the Editor interpolate command. The status code must be a single character in the range J - Z or 0 - 8. Default status code is J.

An Example Rule file that will enable the SPI rule is below:

```
OUT 2 MAX 5 AVE W1 -2 W2 1 Q 7 S J
OUT 1 MAX 10 STA J
NON 1 MAX 10 STA J
SPI MAX 100 INT STATUS K
```

The rule will be logged in the rules file report in the X210 Validation Report.

Energy Discrepancy Rules

ENErgy 1 — Energy Discrepancy Rule 1

ENErgy 2 — Energy Discrepancy Rule 2

These Rules are applicable to cuts that fail the ENErgy Internal Validation Test.

ENErgy 1

Eligibility:

To determine whether a cut being validated (the "current cut") is eligible for correction under Energy Discrepancy Rule 1, the program examines the discrepancy that was found between the current cut's metered energy and its total interval energy. The cut is eligible if the difference between this discrepancy and the cut's Meter Multiplier is approximately 1, and if the preceding cut in the series, if any exists, has a Meter Stop Reading equal to the Meter Start Reading of the current cut.

The first cut being validated in the series constitutes a special case. When the first cut currently being validated in this series may not be the first of the series in the database (for example, because a DATE Command in the Validation Environment File has excluded earlier cuts from being validated), Client/Server Oracle Utilities Load Analysis will attempt to read the cut preceding it from the database in order to determine the current cut's eligibility for correction under the ENErgy 1 Rule.

Action:

If the eligible current cut's total interval energy is greater than its metered energy, an Editor SET Command is generated to reduce the cut's Meter Start Reading by 1. Otherwise, an Editor SET Command is generated to increase the Meter Start Reading by 1.

ENErgy 2

Eligibility:

To determine whether a cut being validated is eligible for correction under Energy Discrepancy Rule 2, the program examines the discrepancy that was found between the current cut's metered energy and its total interval energy. The cut is eligible if: the discrepancy is equal to an integer value consisting of 9s (9.000, 99.000, 999.000, 9999.000, 99999.000, etc.); the sum of all the cut's interval values is 0 (zero); *and* the cut's Meter Start Reading differs from its Meter Stop Reading by 1.

Action:

An Editor SET Command is generated to set the cut's Meter Stop Reading equal to its Meter Start Reading.

Multiple Sets of Edit Rules

In addition to overall edit rules, you may also use the Edit Rules File to define multiple sets of rules for use with particular cuts. Begin each of these extra sets of edit rules with a BLOck Command:

```
BLOck  n
```

The BLOck Command indicates that all edit rules entered after it (until the next BLOck Command, if any) are to be considered part of edit rule set "n", where "n" is an integer from 1 to 999. You may enter up to 999 BLOck commands, followed by their respective rules; the Block numbers must be in ascending sequence, but gaps are permitted. The rules that can be used in each individual Block are the same as those used to define overall edit rules, described above.

If you use a BLOck Command in the Edit Rules File, then for each cut considered for automatic editing, the Validation Program will search the Auxiliary Database (AXDB) for a Validation (type V) record with that series' key. If such a record exists and the number in its text field matches the "n" on any Edit Rules File BLOck Command, the rules in that Block will be applied to that cut. If a cut or its series has no type V record in the AXDB, or has one that specifies a number other than one defined on a BLOck Command, it will be subject to the overall edit rules (specified before the first BLOck Command). (*Note:* The AXDB is described in *Chapter 7: Using the Auxiliary Database (AXDB) to Modify Incoming Data (X170, X180)*.)

If the first Edit Rules File Command *is* a BLOck Command, there are no edit rules for a cut series without a type V AXDB record, and cuts in such series will not be subject to automatic editing.

Automatic Editor Outputs

Execution reports — The *Validation Edit Rules Report* lists the Edit Rules in effect for the Validation run, along with an explanation of each rule; it also lists error messages for any invalid rules found in the Edit Rules File. The *Validation Execution Log* indicates how many Load Data Editor commands, if any, were generated for each cut validated. If any Editor commands were generated, the *Validation Run Summary* indicates the total number of these and the number of cuts affected. Samples of these reports can be found at the end of this chapter.

File containing generated Load Data Editor commands (TGX31AE) — This file is passed to the Load Data Editor for execution of the edits. Preceding each Editor Action Command, a REMark Command is included to indicate that the edit was performed by the Automatic Editor, and the nature of the problem corrected.

Return code — If any Editor commands are generated by the Validation Program, the program sets a return code of 5. This code indicates to Oracle Utilities Load Analysis that the Load Data Editor is to be run to execute the generated edits, followed by a second execution of the Validation Program to re-validate the edited cuts.

Sample Edit Rules File

This Edit Rules File produced the Edit Rule Report at the end of this chapter.

```
NON  1  MAX  4  AVE
NON  2  MAX  9  AVE  AD  W1  -2  W2  1  Q  '7'  S  'K'
OUT  1  MAX  10  INT
BLO  1
ENE  1
ENE  2
NON  1  MAX  1  STA  N
NON  2  MAX  10  AVE  AW  W1  -3
OUT  1  MAX  5  AVE  AW  W2  3  Q  '7'  S  'K'
OUT  2  MAX  1 STA  W
```

Chapter 11

Validation Statistics Reporter

The Validation Statistics Reporter is an extension of the Oracle Utilities Load Analysis System designed to help electric utilities in the reporting of invalid cuts of data that have been loaded by the input program. The statistics from this report can be used to indicate data quality on each translated input file and provide a benchmark for estimating required editing time. It examines the results of four internal validation tests and summarizes the number and percentage of cuts failing each of the validation tests. The four tests evaluated are:

- An INTERVAL test to check the number of intervals found on a cut against the number of intervals calculated from the meter-start and meter-stop times
- An ENERGY test to check meter energy against interval energy
- A NON-NORMAL test to check the number of non-normal status codes found against a user supplied parameter
- An OUTAGE test to check the number of intervals with a status code of "1" against a user specified tolerance.

The Validation Statistics Reporter is typically inserted into the Load Data Input procedure after Load Data Validation. It also produces a report summarizing the results of internal validation for selected cuts. The report indicates the number of cuts and the percentage of cuts that failed each internal validation test and certain combinations of internal validation tests. It then provides a summary of the number of cuts included in the report and the percentage of those cuts that were internally invalid.

Topics covered in this chapter are:

- **Validation Statistics Reporter Inputs**
- **Validation Statistics Reporter Processing**
- **Validation Statistics Reporter Outputs**

Validation Statistics Reporter Inputs

The Validation Statistics Reporter requires two inputs:

- The Current Load Database (CLDB)
- A Control File.

The Control File description follows.

The Control File contains a list of request keys of CLDB cuts that are to be included in the report. There is one request key per line; commas and blanks can separate parameters. A request key has the format:

customer-id, channel, start-time

where:

customer-id — is the customer-id of a cut to be included in the report

channel — is the channel number of the cut

start-time — indicates a specific start-time for the cut. The start-time can be either the "mmddyyhhmmss" or "mm/dd/yy-hh:mm:ss" form.

This input does not have to be manually entered when the program is included in the production input procedure. The Control File is produced by Load Data Input as an output file.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Validation Statistics Reporter Processing

The Validation Statistics Reporter reads the Control File and locates the requested record in the CLDB. The CLDB record, having already been through the validation program, contains fields indicating the internal validation test(s) that have not been passed. The program examines these fields, increments the appropriate program counters relating to each validation test, and reads in the next Control File record to be processed. Once the Control File has been completely processed, a report is generated summarizing the number of cuts that failed to pass the internal validation test(s) performed.

Validation Statistics Reporter Outputs

The Validation Statistics Reporter produces two output reports:

- Internal Validation Error Log
- Internal Validation Summary Statistics Report.

The Internal Validation Error Log lists any requested cuts that were not found in the CLDB. The Internal Validation Summary Statistics Report summarizes the type and number of internal validation tests failed by cuts included in the Control File.

Chapter 12

Using the Time Series (X400) and Load Data (X410 and X420) Reporters

In addition to the reporting capabilities built into the analysis programs, Oracle Utilities Load Analysis offers three separate Load Data Management System (LDMS) programs for reporting data in the interval databases:

- **Time Series Reporter** — reports user-specified groups of statistics or load data records in a tabular (columns and rows) format. Data for each selected record is reported in a single column; each row represents a different hour or time period. This program is especially useful for comparing different statistics across time—even those produced by different analysis programs. An optional file can be produced for input to user-written programs.
- **Load Data Reporter** — Produces reports on individual cuts, cut series, or subsets of a cut series. You would use this program to investigate cuts before editing or to generate detailed information in response to specific requests from customers, other departments, etc. This chapter explains how to use the Load Data Reporter.
- **Summary Reporter** — Summarizes the entire contents of the interval databases. This program is useful for examining the database and investigating problems. *Chapter 13: Using the Summary Reporter for the CLDB or ALDB (X440 or X460)* explains operation of the Summary Reporter.

If you have used the Load Data Input, Direct Input, or Editing procedures, you have already seen some of the reports produced by the Load Data Reporter. This chapter explains how to select cuts for reporting, and how to tailor the reports to your specific requirements.

Time Series Reporter (X400)

For information regarding the Time Series Reporter, see the description of the Y410 Time Series Reporter Program in *Chapter 12* of the *Oracle Utilities Load Analysis Load Data Analysis User's Guide*.

The Load Data Reporter

The Load Data Reporter produces detailed reports on individual cuts, cut series, or subsisted portions of cut series in the CLDB, ELDB, SLCB, and/or ALDB. The Reporter is very flexible and gives you many options for printing, viewing, and graphing detailed information on customers:

- **Load Data Report** — Includes descriptive information, storage and validation flags, validation messages, and edit trails.
- **Data Dump Report** — "Dumps" of interval data for the specified cut or cut series — i.e., a complete list of every interval value. You can specify interval data in two formats:
  - *interval or energy dumps* — actual intervals as they are stored in the database.
  - *demand dumps* — interval values multiplied by the number of recording intervals per hour.
- **Summary Reports** — Summaries of peak, minimum, or daily data. You can request three types of summary reports:
  - *Peak Summary Report* — displays from one to 50 of the highest energy values in the cut, series, or subset, with their times
  - *Minimum Summary Report* — displays from one to 50 of the lowest energy values in the cut, series, or subset, with their times
  - *Daily Energy, Peak and Minimum Report* — total energy, peak, and minimum values for each day in the cut, series, or subsets.

Samples of these reports are shown at the end of this chapter.

Special Note about the Load Data Reporter

Keep in mind that the data dumps can be very lengthy and use a lot of paper if printed. Be careful to request and print only those dumps that you really want.

Steps for Using the Load Data Reporter (X410 or X420)

The following pages describe the steps for using the Load Data Reporter (Figure 12-1 and the command list).

Summary — Using the Load Data Reporter

1. Identify customers to be reported and determine type of report needed for each customer.
2. Create Environment File (TGX41B.ENV).
3. Create Control File (TGX41A.CTL).
4. Verify or modify other input files:
   - Holiday File (TGY31C.HOL)
   - Time-Of-Use File (TGY31D.TOU)
   - Season File (TGY31E.SEA)
5. Run Load Data Reporter (X410 for CLDB, X420 for ALDB, or X410 for both).

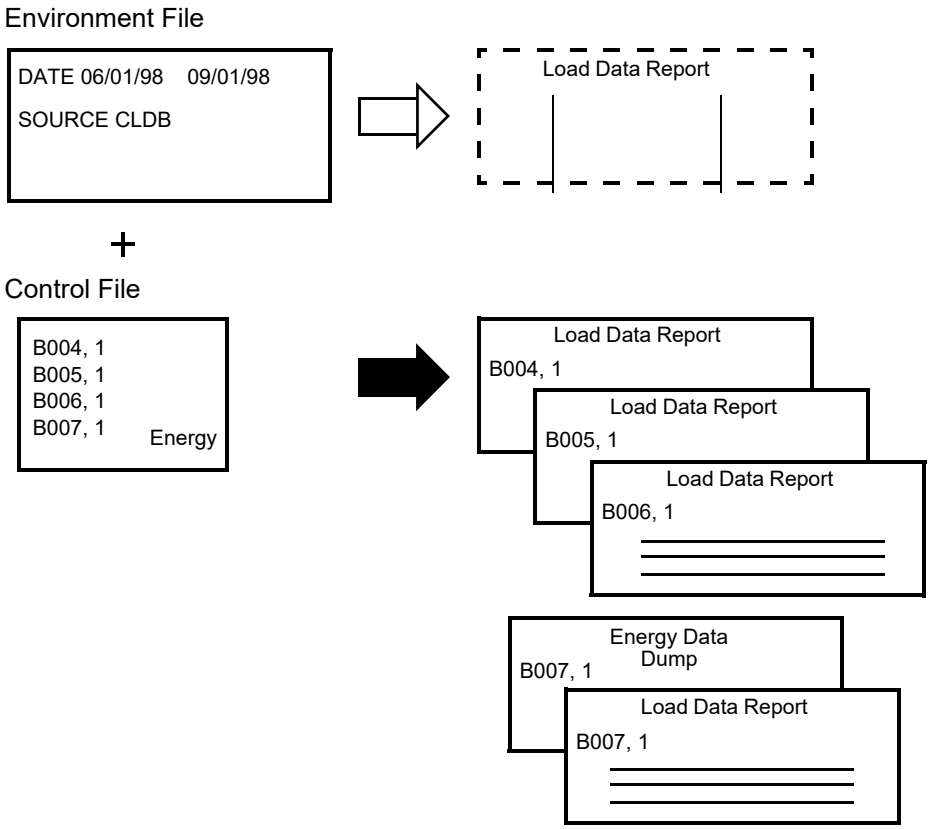


Figure 12-1 Figure 12-1 Constructing Load Data Reports

Create the Environment File with the following commands:

DATe [*start-time*] [*stop-time*]

SOUrce [**CLDB** | **ALDB**] [**BOTH**] [2]

SUBset [**YES** | **NO**]

AGGregate [*n* | 3600]

ROLling [*n* | 3600 [‘*q*’ | ‘8’]]

ORIginal [**INActive** | **Active**]

SCHedule [*n* | 0]

SEAson [*s* | 0 [,**PRInt** | ,**NOPrint**]]

DEMand [**SPReadsheet** | **NOSpreadsheet**] [**NOReport** | **REPort**]

ENErgy [**SPReadsheet** | **NOSpreadsheet**] [**NOReport** | **REPort**]

PEAk

MINimum

DAIly

SUMmary

NUMber [*n* | 10]

MERge [**YES** | **NO** | **EXClude**]

XML

GRA

RUN STA <*un-delimited list of status codes*>

RUN INT [<*value*> | <*lower*> TO <*upper*>]

RUN INS <*interval value*> <*status code*>

Step 1: Identify Customers to be Reported and Types of Reports

The flexibility of the Reporter enables you to quickly generate reports for a group of customers in a single session. The report type can be the same for all customers, or specially modified for each one.

Step 2: Create Environment File (TGX41B.ENV)

You define the contents and format of the reports using a combination of the Environment File and the Control File. In general, use the Environment File to define the basic report (in other words, the report structure you will want most often). Use the Control File to specify the customers to be reported, and any modifications of reports required for individual customers. Remember, commands in the control file will override the environment file commands just for that request.

DATe [*start-time*] [*stop-time*]

- **Date** — Use this command to identify the date range of the reports, where “start-time” specifies the beginning boundary of the date range and “stop-time” specifies the ending

boundary of the date range. Those cuts specified in the Control File whose Start Times fall within the date range will be reported. You can use either the "mm/dd/yy-hh:mm:ss" or "mmddyyhhmmss" format. If you specify a start-time but no stop-time, the stop-time defaults to infinity. If you supply neither time, the DATE Command has no effect on requests.

> **Important note:** Whether the program reports individual cuts or reports merged data for a user-specified segment of time in a cut series depends upon how you set up the Date and Subset commands, as well as whether or not you specified cut keys or cut series keys in the Control File. This has an impact on not only the date range of the interval data in data dumps, but also on meter readings and energy values in the descriptive portion of the reports.*Table 12-1 details how you can set up the appropriate commands and keys to achieve the desired results.* (Remember, a "cut" represents energy use over an actual recording period in the field; Oracle Utilities Load Analysis merges cuts into "cut series" that represent continuous, on-going measurement of energy use.)

SUBset [**YES** | **NO**]

- **Subset** — An optional command. In general, specify "YES" if you want to produce one report for the portion of each cut series that falls between the two dates you specified in the Date Command. ***Note:*** If you use the Subset Command, you must specify *both* a start- and stop-time in the Date Command. Otherwise, the program will issue an error message and terminate processing. See Table 12-2 for additional details about the use of the Subset and Date commands.

SOUrce [**CLDB** | **ALDB**] [**BOTH**] [2]

- **Source** — Use this command to specify the origin of the data you wish to report. "CLDB" **must** be specified with this command if you are using the CLDB Load Data Reporter (X410), "ALDB" if you are using the ALDB Load Data Reporter (X420). To report data from both the CLDB and an ALDB, specify "CLDB BOTH" and use X410, the CLDB Load Data Reporter. After either "ALDB" or "CLDB BOTH", you may add the optional parameter "2" if you wish to include data from a second ALDB as well.

AGGregate [*n* | 3600]

- **Aggregate** — An optional command. Use "n" to specify the seconds-per-interval for reporting interval dumps. If possible, load data will be summed or averaged. Units of measure and their corresponding aggregation methods (summation or averaging) are shown in *Appendix A: Oracle Utilities Unit of Measure Codes*. For example, if you input "AGG 1800", each interval would represent 1800 seconds (30 minutes). The default is no aggregation — the intervals would be reported in their current seconds per interval.

ROLling [*n* | 3600 ['*q*' | '8']]

- **Rolling** — This command is most often used for billing purposes, and can be found in rate schedules. This command smooths out demand and reports averages of intervals. Use "n" to specify the interval length you want to average (the result is called the "rolling interval"). "n" must be greater than, and a multiple of, the cut's interval length. For example, let us say that your data exists in 900-second (15-minute) intervals and you have consecutive values of 8, 9, 8, and 16. If you made n=1800 and applied the ROLling option, the 16 would become 12; that is, (8+16) / 2=12. Use "q" to indicate the worst acceptable status code involved in the averaging. Often rates for large customers contain a rolling demand clause, and this option is useful in those circumstances.

ORIginal | **INActive** | **Active**

- **Original or Active** — Use this command to indicate which cut version you want to report. Specify "ACT" if you want to report the "active" version — that is, the only version that exists for an unedited cut, or the latest version of a cut that has been edited. Specify either "ORI" or "INA" if you want to report the original, "inactive" version of an edited cut.

  Note: ORIginal and INActive apply only to cuts that have been edited. The default is ACTive.

SCHedule [*n* | 0]

- **Schedule** — An optional command. Use it to specify that data be reported by entire period and time-of-use periods in the "Peak Summary", "Minimum Summary", and "Daily Energy, Peak and Minimum" reports. (Your utility's time-of-use periods are specified at system setup in the Time-of-Use File—see *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*.)

  Use "n" to specify the Time-of-Use Schedule you want used. The default is "0", which means that just the entire period will be reported..

SEAson [*s* | 0 [,**PRInt** | ,**NOPrint**]]

- **Season** — An optional command. Use it if you want to apply different Time-of-Use schedules to different portions (seasons) of the reporting period. (Your utility's season schedules are specified at system setup time in the Season File — see *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*.) It is very important to keep in mind that any schedule you select must cover the entire reporting period. If the program encounters a gap, it will abort processing and issue an error message.

  Use "s" to specify the Season Schedule in the Season File that you want used. The default is "0", which means no Season Schedule will be used. If you input a Season Schedule and you want to see the combined time-of-use graph that results from its application, specify "PRINT"; the default is "NOPRINT".

  Note: Use either the SEASON Command or the SCHEDULE Command to define time-of-use periods for the daily and min/max reports, but not both. If you include both commands in the Environment File, the program will use the SEASON Command and disregard the SCHEDULE Command. If you include neither command, the default is "SCHEDULE 0, NOPRINT".

SEParate

- **Separate** — An optional command when using the Season Command. Input the keyword "SEParate" (with the PEAk and/or MINimum commands, in the Environment File) to produce a separate Peak Summary and/or Minimum Summary report — a list of the one to 50 highest/lowest energy values in the cut or cut series, with times — for each season.

DEMand [**SPReadsheet** | **NOSpreadsheet**] [**NOReport** | **REPort**]

- **Demand** — An optional command. Input the keyword "DEMAND" if you want the report to include a complete listing of actual intervals in demand units ([intervals] x [# of intervals per hour]). The optional SPReadsheet and NOReport parameters enable you to select whether this list will be produced in report format, as a comma-delimited file (suitable for use in a spreadsheet program), or both. The report format is produced by default (specify NOReport to override), and the spreadsheet is *not* produced by default (specify SPReadsheet to override).

ENErgy [**SPReadsheet** | **NOSpreadsheet**] [**NOReport** | **REPort**]

- **Energy** — An optional command. Input the keyword "ENErgy" if you want the report to include a complete listing of actual intervals in energy units. The optional SPReadsheet and NOReport parameters enable you to select whether this list will be produced in report

format, as a comma-delimited file (suitable for use in a spreadsheet program), or both. The report format is produced by default (specify NOReport to override), and the spreadsheet is *not* produced by default (specify SPReadsheet to override).

PEAk

- **Peak** — An optional command. Input the keyword "PEAk" if you want to produce the "Peak Summary Report" — a list of the one to 50 highest energy values of the cut or cut series, with times. The default number of peaks to report is 10; the maximum number of peaks to report is 50.

MINimum

- **Minimum** — An optional command. Input the keyword "MINimum" if you want to produce the "Minimum Summary Report" — a list of the one to 50 lowest energy values in the cut or cut series, with times. The default number of low values to report is 10; the maximum number of low values to report is 50.

DAIly

- **Daily** — An optional command. Input the keyword "DAIly" if you want to produce the "Daily Energy, Peak and Minimum Report" — a list of the total energy, peak, and minimum values for each day in the cut or cut series.

SUMmary

- **Summary** — An optional command, equivalent to specifying PEAk, MINimum, and DAIly. Input the keyword "SUMmary" if you want all three of these reports.

  Note: Any of the three Summary reports requested in the Environment File cannot be turned off for individual customers in the Control File.

NUMber [*n* | 10]

- **Number** — An optional command, used to specify the number of peaks and/or minimums reported for each cut on the Peak Summary and Minimum Summary reports. The default is 10 and the maximum is 50. The NUMber Command should be input only once for both Peak Summary and Minimum Summary reports.

MERge [**YES** | **NO** | **EXClude**]

- **Merge**— An optional command. When set to "YES", the program will merge all cuts found within the specified date range regardless of validation status. When set to "NO" (default), the program will reject invalid cuts and return an error message indicating the cuts are not merge-able. When set to "EXClude", the program will merge all cuts found within the specified date range, but exclude all invalid data.

XML

- **XML** — An optional command. When present, all reported cuts will be exported to a standard Oracle Utilities Interval Data XML file located in the job folder..

GRA

- **GRA** — An optional command used to produce the SYSGRAPH.HTM file, also in the job folder, which can be used to graph the reported cuts.

RUN STA <*un-delimited list of status codes*>

- **RUN STA** — An optional command used to search and report on the cuts with runs of specified status codes.

RUN INT [<*value*> | <*lower*> TO <*upper*>]

- **RUN INT** — An optional command used to detect interval value runs within the cut or date range of intervals. It has two basic modes of operation, Single and Range. Single mode allows users to search for runs of a single-value interval within the cut of data. For example, if a user would like to search a cut for interval values of 9, or 0.
- Range mode allows users to search for a range of intervals in a run such as 0 TO 10. Runs will be identified by any interval value between and including the lower and upper value provided.

RUN INS <*interval value*> <*status code*>

- **RUN INS** — An optional command used to report on a specific interval value and a specific status code. For example, if a user wants to report runs of interval value 0 and status code 8.

Step 3: Create Control File (TGX41A)

The Control File is made of your report requests. Each request consists of the key for the customer you want to report, and any modifications or additions to the reports required for that customer. In each request, you can specify a single cut or a cut series. You specify a cut by supplying the Full Key; i.e., customer-id, channel-number, start-time. You specify a cut series by supplying just customer-id, channel-number.

Remember, the options specified in the Environment File become the default options to be used, unless reset on a cut-by-cut basis in the Control File. The options capable of being overridden in the Control File are AGGREGATE, ROLLING, ORIGINAL, SCHEDULE, SEASON, and NUMBER. Options overridden in the Control File are only in effect for the cut requested, and revert back to the default options of the Environment File for the next Control File cut. The options ENERGY and DEMAND specified in a cut request of the Control File add the corresponding detailed data dump to the detailed data dumps requested in the Environment File, but only for the requested key. A detailed data dump request made in the Environment File *can be turned off in the Control File* by use of the appropriate "NO" option (NOReport to turn off the conventional report, NOSpreadsheet to turn off the spreadsheet) following the ENERGY or DEMAND options.

If the Environment File has the following options:

- AGGregate 3600
- ENErgy
- ORIginal

and the Control File record specifies:

- AGGregate 0
- DEMand
- ACTive

then the active record will be reported with no aggregation and both *energy* and *demand* dumps will be printed. Table 12-1 demonstrates the outcome of some combinations of Environment File and Control File options. Please note that DATE RANGE, SUBSET, and SOURCE may only be specified in the Environment File.

Table 12-1: Sample Options Combinations

AGGREGATE		ROLLING		SCHEDULE		TYPE		DUMPS		RESULTS
ENV	CON	ENV	CON	ENV	CON	ENV	CON	ENV	CON	
3600		0		0 NOP		ACT		ENE		aggregation 3600, active, energy
3600	0	0	1800	1 PRI	2NOP	ACT	ORI	ENE	DEM	no aggregation, rolling, 1800, schedule 2, no graph, original, energy, demand
3600		0	1800	1 PRI		ACT		ENE		error: roll factor 1800 <= aggregation factor 3600

To create the Control File, you will use the following format. Input one request per line. Remember, you need to specify bracketed commands only when you want something other than what was specified in the Environment File.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Although the commands shown below take up several lines, each one of your requests must fit on a single 200-character line. The descriptions of the commands are the same as those provided in Step 2.

customer-id, *channel* [,*start-time*] [,**ENErgy**] [,**DEMand**] [,**ORIginal** | **INActive** | **ACTive**] [,**SCHEdule** [*n* | 0]] **SEAson** [*s* | 0 [**PRInt** | ,**NOPrint**]] [,**AGGregate** [*n* | 0]] [,**ROLling** [*n* | 3600 [‘*q*’ | ‘8’]]] [,**PEAk**] [,**MINimum**] [,**DAIly**] [,**SUMmary**]

- **Customer-id** — Input the customer-id of the cut or cut series you want to report on. *This is a required field.*
- **Channel** — Input the channel number of the cut or cut series you want to report on. *This is a required field.*
- **Start-time** — If you want to report on a single cut, input its start-time (you can use either the “mmddyyhhmmss” or “mm/dd/yy-hh:mm:ss” format). If you do not specify a start-time, Oracle Utilities Load Analysis will use the date-range specified in the Environment File.
- **Stop-time** — Used to override the DAT command in the env file for this cut only. *It must follow the Start-time.*

Here is a sample Control File (Figure 12-2):

```
B001, 1, 07/01/99-
00:00:00  ENE
```

Figure 12-2 Sample Load Data Editor Control File

Some additional sample report requests are shown below:

- Q1111,1 ENERGY DEMAND
  (dumps for all cuts from channel 1)
- A1234 2 12/12/99-11:30:00 ORIGINAL
  (one specific cut, inactive record of original data)
- K1212,1 12/30/99-04:00:00
  (one channel, specific start-time)
- Z2121,1, 10/04/99-10:04:00, AGG 3600, SCH 2 PRINT
  (one channel, specific start-time, data aggregated to 3600 seconds (60 minutes), the Peak and Minimum Summaries and Daily Energy, Peak and Minimum Report use Time-of-Use Schedule 2, prints time-of-use graphs)
- SMITH4,1
  (entire cut series).

A Closer Look at the Interaction Between Environment and Control Files

As stated earlier, the form in which the data will be reported depends on the interaction between the Environment File and Control File options. Table 12-2 presents all the different combinations of Date and Subset Commands and customer key options that you can choose. The following text details the results of each combination.

1. Reports all cuts in the requested cut series whose data, either part or all, falls within the date range given. A detailed data dump includes a full dump for each of the cuts chosen in the series. The associated meter readings and energy values reported reflect the actual cut values.
2. Reports a full cut if the cut start-time falls within the date range given. A detailed data dump only includes data for the specified cut. The associated meter readings and energy values reported reflect the actual cut values.

3. Reports a merged cut consisting of data from all cuts of the indicated cut series whose data falls within the date range specified. Empty intervals of the merged cut are assigned a zero value and a status code of nine ('9') to indicate missing data. The merged cuts' meter readings are set to zero and the reported energy corresponds to the actual energy of the cut. A detailed dump includes data for the merged cut whose start- and stop-times correspond to the start- and stop-times of the date range.

4. Reports one full cut if the cut's start- and stop-times both fall within the indicated date range, or that portion of the cut that falls between the cut's start-time and the stop-time of the date range. The meter readings are set to zero. The reported energy corresponds to the actual energy of the cut's data that falls within the date range. All empty intervals that are in the date range, but not included in the required cut, are zero filled and assigned a status code of nine ('9') for *missing*.

5. Reports all cuts in the cut series that have data that falls after the start-time of the date range. Each full cut is reported separately with all of its accompanying descriptive data.

Table 12-2: Combinations of Load Data Reporter Input Options

		In Environment File				In Control File	
		DATE		SUBSET		REPORT REQUEST	
		START-TIME	STOP-TIME	YES	NO	CUT KEY[a]	CUT SERIES KEY[b]
Valid Combinations	1.	X	X		X		X
	2.	X	X		X	X	
	3.	X	X	X			X
	4.	X	X	X		X	
	5.	X			X		X
	6.				X	X	
	7.	X			X	X	
	8.				X		X
Invalid Combinations	9.		X		X		X
	10.		X		X	X	
	11.		X	X			X
	12.			X			X
	13.	X		X			X
	14.		X	X		X	
	15.			X		X	
	16.	X		X		X	

a. customer-id, channel-number, start-time (also called the "Full Key")
b. customer-id, channel-number
X: denotes the inclusion of the stated option.

1. Reports one full cut corresponding to cut key and start-time requested. All descriptive data corresponds to the individual cut reported.

2. Reports one full cut if the indicated cut's start-time falls after the start-time of the specified date range. All descriptive data reported corresponds to this individual cut.

3. Reports all cuts in the cut series regardless of start-time. Each full cut is reported separately with all of its accompanying descriptive data.

4. Invalid option combination. A stop-time cannot be requested without a start-time. The time on the DATE Command will be interpreted as the start-time of the date range, and the same report will be produced as for option combination (5). The option is incorrect as stated, and the user is warned that the reported results may not be as expected.
5. Invalid option combination. A stop-time cannot be requested without a start-time. The time on the DATE Command will be interpreted as the start-time of the date range, and the same report will be produced as for option combination (7). The user is warned that the reported results may not be as expected.
6. Invalid option combination. A stop-time cannot be requested without a start-time. The time on the DATE Command will be interpreted as the start-time of the date range, and the request will be invalid because *both* start- and stop-time must be specified when subsetting is selected. A diagnostic message is returned to indicate the error.
7. Invalid option combination. A date range must be specified when subsetting is selected. A diagnostic message is returned to indicate the error.
8. Invalid option combination. A a stop-time for the date range must be specified when subsetting. A diagnostic message is returned to indicate the error.
9. Invalid option combination specified. See option combination (11) because the errors are the same.
10. Invalid option combination specified. See option combination (12) because the errors are the same.
11. Invalid option combination specified. See option combination (13) because the errors are the same.

Step 4: Verify or Modify Other Required Input Files

Along with the Control and Environment files, three other input files may be required by the Load Data Reporter (depending upon your selections in the Environment File):

- Holiday File (TGY31C.HOL) — a list of holidays observed in your service territory.
- Time-Of-Use File (TGY31D.TOU) — a series of TOU schedules.
- Season File (TGY31E.SEA) — a series of season schedules that enable you to apply different TOU schedules at different times of the year.

Typically these files are established at system installation and seldom modified. Should you need to update or modify these files, see the instructions provided in *Chapter 5: Setting Up Your Oracle Utilities Load Analysis System*. Because these files are used by several Oracle Utilities Load Analysis programs, they should be modified only under special circumstances and according to the policies of your utility.

Step 5: Run the Load Data Reporter (X410 or X420)

Once you have created the environment file and the control file you are ready to run the Load Data Reporter Procedure. You may override the default files by selecting alternate files located in your DATA directory.

Load Data Reporter Processing

The Load Data Reporter Program checks the Environment File to determine what options have been chosen and if they are consistent. The following conditions halt processing:

- Subset mode specified without a date range or with an incomplete date range (i.e., a start-time and no stop-time)
- An invalid aggregation level (i.e., a level that is lower than the data's original level of aggregation)
- An invalid rolling level of aggregation (same as aggregation problem above)
- A rolling level of aggregation equal to the data's original level of aggregation
- Subset specified and cuts not eligible for merge. (May be overridden by MER command.)

Each key in the Control File is processed by finding a corresponding cut or cut series in the CLDB or ALDB. When a date range is specified in the Environment File, if the start-time of the request key (if stated) is not within the date range, an error message is produced and processing continues with the next key in the Control File. If subset is set to "YES", one report is generated that includes data from one cut (if the request key is a full key), or from a cut series whose data falls between the date range specified. If subset is set to "NO", a report for each cut that has data within the date range is generated, or one report is produced for a Full Key request. ***WARNING:*** When data is aggregated, the original start- and stop-times of the CLDB or ALDB cut are used for comparison with the date range. A stop-time of the reported cut is calculated using the number of recorded intervals, and may differ from the original cut stop-time if data is missing.

If date range is set to "ALL", data for all cuts in the CLDB or ALDB with the same customer-id and channel as the request key is selected, and a descriptive report is generated for each cut. If the request key is a Full Key, one report for one cut is produced.

If aggregation and/or subsetting are requested, the stop-time, recorded and expected intervals, interval energy, and total energy of the reported cut are calculated from information from a CLDB or ALDB cut or cuts; *meter data is set to zero. Printing of the validation messages is suppressed* because these messages do not apply to the synthesized cuts reported. If at least one of the options "Energy" or "Demand" is specified, interval data report(s) follow the descriptive data report. ***WARNING:*** If rolling is specified and if the status code of each interval is less than or equal to the worst acceptable status code stated in the Environment File, the interval is not included in the average.

If the SCHEDULE Command is included in the Environment File and a schedule number other than zero is chosen, any Peak Summary, Minimum Summary, and Daily Energy, Peak and Minimum reports requested will be broken down by time-of-use periods associated with that schedule number. If the "PRINT" option is chosen, time-of-use period graphs will be printed as part of the output; otherwise these are omitted.

Chapter 13

Using the Summary Reporter for the CLDB or ALDB (X440 or X460)

With the Summary Reporter, you can get a hard copy listing of the entire contents of the interval databases, or of a specified portion of the database. These reports can be very useful if you encounter problems with the Oracle Utilities programs. In fact, before you call the Oracle Utilities Help Line, you should run this program.

Caution should be used when submitting this program against large databases. The output may become unmanageable.

The Summary Reporter is an easy-to-use program, and one that yields very useful information. You can get a hard copy printout (if you choose to print) of the entire contents of the interval databases, of selected cut series, and/or of all cuts within a specified time period. The Summary Reporter will also generate important statistics — including the total number of cuts; the number of valid and invalid cuts; the number of active records, inactive records, and edit trails; and the number of cuts with forced Merge and Archive fields.

Steps for Using the Summary Reporter (X440 or X460)

The following pages describe the steps for using the Summary Reporter with either the CLDB or ALDB Summary Reporting Programs.

Summary — Using the Summary Reporter

1. Determine whether you want a report on the entire database or on selected cuts. If you want a report on the entire contents of the database, go to Step 3. If you want to report only selected cuts, create the Control File (TGX44A.CTL) — a list of cut series keys.
2. Create the Environment File (TGX44B.ENV) — optional reporting period and other parameters.
3. Run the Summary Reporting Program.

Step 1: Create the Control File (TGX44A.CTL) to select specific cuts for reporting.

If you want to report the entire contents of the database, go to Step 3. The Control File (Figure 13-1) is a list of the cut series you want to report. You can create the file manually or with the CLDB or ALDB Key Generator Program.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Each report request has the following format:

```
Customer-id, channel-number
```

Enter one request per line. You **must** enter the requests in ascending order (A - Z; 1, 2, 3, etc.). You can separate the two elements of a request (customer-id and channel-number) with blank spaces and/or a comma.

If you do not supply a Control File, the program will consider all cuts in the database eligible for reporting.

> **Note:** If you do supply a Control File, it will only be used if you specify SELECT KEY in the Environment File.

```
B001, 1
B001, 2
B003, 1
B004, 1
B004, 2
B005, 1
```

Figure 13-1 Sample Control File

Step 2: Create the Environment File (TGX44B)

Use the Environment File (Figure 13-2) to specify an optional date range for the reporting period, and other parameters.

Create the Environment File with these commands:

DATe [*start-time stop-time* | **ALL**]

SELect [**KEY** | **ALL**]

SOUrce {ALDB | **CLDB**}

[**NOEdit** | **EDIT**]

MAXimum

FACtor

HOUrs

OUTages

SPIkes

DIPs

- **Date** — Use this optional command to set the start- and stop-time of the reporting period for the *selected cuts*. You define "selected cuts" using the Select Command described below — either all cuts in the database or just those listed in the Control File.

  There are several ways to specify the reporting period:

  - If you specify just ALL, then all selected cuts will be reported, regardless of their start-time. (This is the default.)
  - If you supply both a start-time and a stop-time, all selected cuts with intervals at or between the two times will be reported.
  - If you supply just a start-time, all selected cuts with intervals at or after the specified start-time will be reported.
  - If you specify ALL followed by a stop-time, the program will report all selected cuts with intervals at or before the specified stop-time.

```
DATE
ALL
SELECT
KEY
```

Figure 13-2 Sample Environment File for the CLDB Summary Reporter

To specify a date/time, use the "mm/dd/yy-hh:mm:ss" format. If you omit the "hh:mm:ss" portion, start- and stop-times default to 00:00:00 and 23:59:59, respectively.

SELect [**KEY** | **ALL**]

- **Select** — Use this command to select the cut series you want reported. "ALL" indicates that you want every cut in the database that falls within the specified date range to be reported. "KEY" indicates that you want only those customers whose keys appear in the Control File and whose start- or stop-times fall within the desired date range to be reported. ALL is the default.

SOUrce {ALDB | **CLDB**}

- **Source** — Use this command to identify which format you want the cuts reported in. This is a required command. Specify "CLDB" if you wish to execute the CLDB Summary Reporting Program (X440) and have the data reported in the CLDB format. Specify "ALDB" if you wish to execute the ALDB Summary Reporting Program (X460) and have the data reported in the ALDB format. If you wish to report data in the ELDB or SLDB format, see *Chapter 12* in the *Oracle Utilities Load Analysis Load Data Analysis User's Guide* for specifics.

  [**NOEdit** | **EDIT**]

- **Edit/Noedit** — Use this optional command to determine if the Edit Trail output file will be produced. NOEDIT causes the file not to be written, and EDIT causes the file to be produced. If it is omitted, the file is produced.

  Note: The example for the ALDB Summary Reporting Program is identical, except "SOURCE ALDB" is used.

- **Custom Summary Report commands** — Optionally, you may request a custom report showing additional information *for load data cuts only (not statistics)* by specifying one or more of the following commands:

  MAXimum

  MAXimum — Reports the Maximum Demand for any interval in each cut listed, with the demand unit-of-measure.

  FACtor

  FACtor — Reports the Load Factor for each cut:

$$\text{Load Factor} = \frac{\text{Cut Total Energy}}{\text{Maximum Demand} \times \text{\# of Hours}}$$

  HOUrs

  HOUrs — Reports Hours Use for each cut:

$$\text{Hours Use} = \frac{\text{Cut Total Energy}}{\text{Maximum Demand}}$$

  OUTages

  OUTages — Reports any Outages noted in each cut's Validation Messages.

  SPIkes

  SPIkes — Reports any Spikes noted in each cut's Validation Messages.

  DIPs

  DIPs — Reports any Dips noted in each cut's Validation Messages.

  Note: Outages, Spikes, and Dips are reported only for cuts in which they have been found by the Validation Program (X210). If you are reporting on a database containing only cuts that have not been validated, each of these three options will produce only a column of blanks on the Custom Report. *See Chapter 5 for details.*

Step 3: Run the Summary Reporter

Once you have created the desired inputs, you are ready to submit the job. Use X440 for the CLDB or X460 for the ALDB.

Summary Reporting Processing

When "SELECT ALL" mode is used, each cut on the database that falls within the desired date range is reported.

When "SELECT KEY" mode is used, only those cuts whose keys appear in the Control File and whose start- and stop-dates are within the desired date range are reported. If a key is not found, an appropriate error message is generated and processing continues with the next Control File record.

Chapter 14

The Totalizing Reporter

The Totalizing Reporter is a comprehensive software tool developed to help utilities report and write results on multiple channels of data.

Topics covered in this chapter are:

- **What Is the Purpose of the Totalizing Reporter?**
- **Totalizing Reporter Mechanics**
- **What Input Files Are Used In the Totalizing Reporter?**
- **What Output Files Does the Totalizing Reporter Produce?**
- **Naming Conventions In the Totalizing Reporter**
- **Steps For Using the Totalizing Reporter (X430 and Y450)**
- **Summary Statistics Output File (TGX432) Record Descriptions**
- **Sample Applications of the Totalizing Reporter**
- **Example A — Power Billing**
- **Example B — End Use Study**

What Is the Purpose of the Totalizing Reporter?

The Totalizing Reporter produces specially formatted reports for time-series data stored in Oracle Utilities Load Analysis. In addition to generating tabular data displays, the Totalizing Reporter also produces subtotals and totals of load values for designated customer groups. Summary information such as peaks, coincident peaks, averages, total energy, load factors, and measures of completeness are printed automatically. The user has full control over a title line, remarks, the number of peaks printed, and whether or not status codes are to appear in the report. The user may also control the date range to be reported, time-of-use periods to be reported, the aggregation level of the input data, the detail of the output reported, and whether or not invalid data should be included in the report. The aggregated data may even be saved for use in subsequent analyses.

The Totalizing Reporter is especially useful for applications such as power billing and wholesale delivery. In a situation where several channels of data are recorded for a single customer, a report can be produced to display each channel, subtotals of several channels, and the combined total for each interval of time. The totals and subtotals for each interval may be stored indefinitely allowing the user to maintain a complete load data history for that single customer. Any billing questions are quickly resolved simply by retrieving the total data.

The Totalizing Reporter is also valuable for load studies involving multiple points of delivery. Up to 500 channels and subtotals of data can be displayed in parallel, interval-by-interval. Multiple customers or substations can be accumulated, including cogenerators. Coincident demand and peak information is easily obtained for any desired group of data channels.

Totalizing Reporter Mechanics

This section is intended primarily for anyone unfamiliar with the basic concepts of input and output files. However, it also defines some important terms unique to Oracle Utilities Load Analysis, such as "Environment files" and "Control files." Topics covered in this section are:

- What Input Files are Used in the Totalizing Reporter?
- What Output Files Does the Totalizing Reporter Produce?
- Naming Conventions in the Totalizing Reporter.

Before you begin working with The Totalizing Reporter, you might want to review some of the basics of working with the Oracle Utilities Load Analysis programs.

You will accomplish tasks by submitting programs. Each program requires input files and produces output files. Output files are the results, including reports and data files.

Another way of looking at this is that you can accomplish the tasks by following three main steps:

1. Creating Input Files
2. Running the Program
3. Checking Output Files.

What Input Files Are Used In the Totalizing Reporter?

The Totalizing Reporter uses four types of input files: *Environment File, Control File, Time-of-Use Schedule File,* and *Holiday File.*

- **The Environment File** contains your processing instructions to the program. Typically, you select among a set of optional parameters to define the file.
- **The Control File** typically contains the data to be processed. Customer lists and lists of keys for load data records or statistics are common kinds of control files.

- **The Time-of-Use Schedule File** contains a list of each hour in a week as on-peak, off-peak, or shoulder.
- **The Holiday File** contains a list of dates of all national and local holidays that apply to your service territory.

Note: This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

What Output Files Does the Totalizing Reporter Produce?

Output — the result of running a program — can be new or modified data entered into a database, or reports. Aside from special reports, several reports are automatically produced by the Totalizing Reporter to help you evaluate the success of the program run:

- **Block Summary** describes the processing mode in effect during the program run. Use this report to verify that you entered the correct commands and parameters in the Environment File, and ran the correct control file. Also, any comments used in the block are reported.
- **The Execution Log** typically lists data that was altered or created during the program run; it often includes diagnostic messages which alert you to any problems with the input data or the run.
- **Generated Reports** consist of detailed interval demands and summaries of peaks, averages, energy, and load factor.

In addition, an output file is created, as well as records that can be written to the database:

- The **Summary Statistics Output File** is a comma-delimited file suitable for use by many spreadsheet programs.
- **Load Data Records** may be written back to the CLDB or ELDB for each total or subtotal that is created and reported.

Naming Conventions In the Totalizing Reporter

Elements of the Oracle Utilities Load Analysis program are identified by a consistent set of naming conventions. Being able to recognize programs, procedures, and files by these naming convention will be a big help to you in working with the system. Here are the conventions used in the Totalizing Reporter Extension:

Element	Convention		
Control file	TGX43A.CTL	=	Totalizing Reporter Control File
Environment file	TGX43B.ENV	=	Totalizing Reporter Environment File
Holiday file	TGE31C.HOL		
Time-of-Use file	TGE31D.TOU		
Output file	TGX432.DAT	=	Totalizing Reporter Output File

Steps For Using the Totalizing Reporter (X430 and Y450)

The Totalizing Reporter produces generated reports from load data records in Oracle Utilities Load Analysis's Current Load Database (CLDB) Extracted Load Database (ELDB) or Archive Load Database (ALDB). The Totalizing Reporter requires four input files:

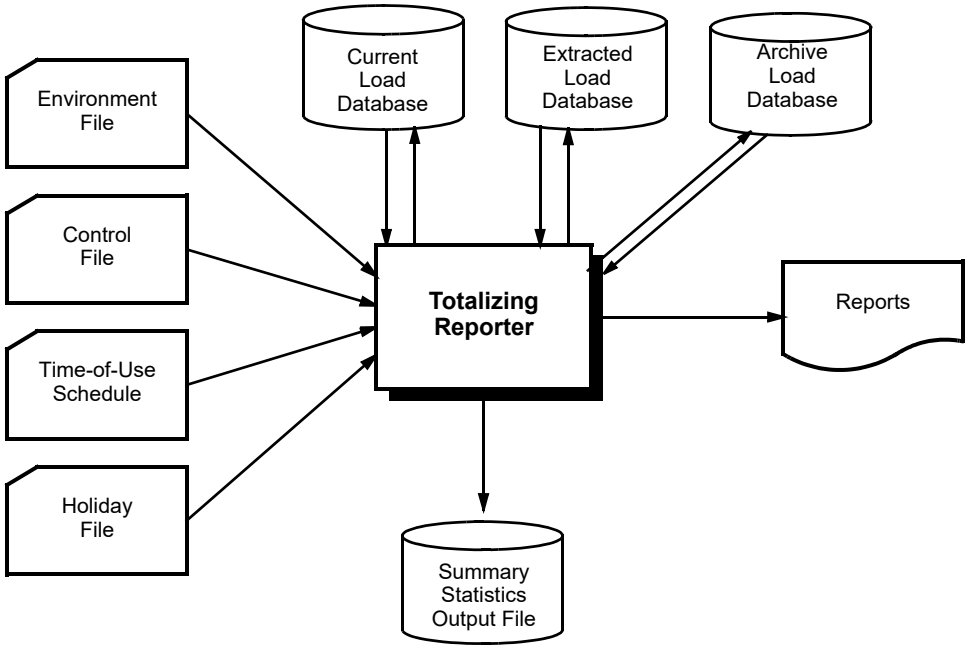
- Environment File
- Control File
- Time-of-Use File
- Holiday File

The Totalizing Reporter may be run on load data in the Current Load Database (CLDB) that has passed validation, on load data that has been extracted to the Extracted Load Database (ELDB), or on data in the Archive Load Database (ALDB).

Here is a brief list of the steps you will follow when analyzing load data with the Totalizing Reporter program. Later in this chapter these steps are explained in detail.

Summary — Using the Totalizing Reporter Program (X430 and Y450)

1. Identify customers to be totalized and determine that they exist on the database you are using (CLDB, ELDB, or ALDB).
2. Create Environment File (TGX43B.ENV).
3. Create Control File (TGX43A.CTL).
4. Modify Time-of-Use Schedule File (TGY31D.TOU) if necessary.
5. Modify/verify that the Holiday File (TGY31C.HOL) has been set up correctly.
6. Run the Totalizing Reporter Program (X430 or Y450).



Step 1: Identify Customers to be Totalized and Reported

The Totalizing Reporter produces formatted reports from load data records in Oracle Utilities Load Analysis's Current Load Database (CLDB), Extracted Load Database (ELDB), or Archive Load Database (ALDB). It is important to identify these records by their cust-id and channel. You also have the option of labeling each record in the report with an alternate identifier such as account number.

Step 2: Create Environment File (TGX43B.ENV)

The Environment File is made up of commands summarized in Figure 14-1. When creating the file, enter one command per line. Each line must begin with the command name (keyword); but you need enter only the first three letters of the keyword. You may enter the commands in any order. Blank lines may be used between commands. Blanks must be used as delimiters between parameters. Comments may be included following "/*" on each command line. If you do not specify a command, the program will assume the default (underlined).

GRA

HIGhest [*n* | 3 [**AVErage** | **AVG**] [**CPK**]]

QUAlity [Q | '8']

REPort [**SUMmary** | **NONe** | **ALL**]

SAVe [**REPlace**] [**ARChive**]

STAtus [**YES** | **ALL** | **NO**]

STOp [**NO** | **YES**]

UOMcheck {**NO** | **YES** [**METhod** | **LEGacy**]} or {**OFF** | **ON**}

VALid [**NO** | **YES**]

Figure 14-1 Format for the Totalizing Reporter Environment File

Following is a detailed description of each Environment File command.

HIGhest [*n* | 3 [**AVErage** | **AVG**] [**CPK**]]

- **Highest** — Use this command to specify the number of interval peaks to report for each customer; "n" is an integer between 0 and 5. The default is "3" if no other options are specified. Specify the Average (AVG) option to generate an average demand value in the summary report. Specify the Coincident Peak (CPK) option to generate the values at the time of the coincident peak in the summary report. If you specify AVErage or CPK, you must also supply a value for "n."

  Note: The largest number of interval peaks that can be accommodated in the report, using the AVG or CPK option is 5. If the options are used together, the number of interval peaks requested cannot exceed 3.

QUAlity [Q | '8']

- **Quality** — Use this command to set the level of acceptable data quality for the Totalizer. Input the worse acceptable status code. Oracle Utilities Load Analysis will treat any interval with a status code worse than Q as missing.

  Note: the status code must be input with apostrophes (') on either side. The default is '8' (only missing intervals are excluded).

REPort [**SUMmary** | **NONe** | **ALL**]

- **Report** — Use this command to limit the Totalizer output. Specify the Summary option to produce only the Summary Statistics Reports. Specify the ALL option to produce both detail and summary reports. The Summary Statistics can be saved to an output file in addition to being reported. See **Summary Statistics Output File (TGX432) Record Descriptions** on page 14-12.

SAVe [**REPlace**] [**ARChive**]

- **Save** — Use this command to specify whether to save a total or a subtotal to the database. If you specify the Replace option, an incoming cut will automatically replace an existing cut in the database that has the same cut key. Specify the Archive option to set the archive flag on for the new cut. The replace and archive options may be used separately or together. Omit this command if no cuts are to be saved to the database. **Note**: Customer-id and channel must be specified in the END command in the control file if SAVE is used.

STAtus [**YES** | **ALL** | **NO**]

- **Status** — Use this command to specify whether status codes should be displayed next to each reported value. Specify the Yes option to indicate that an '*' is to be printed next to any value whose status code is worse than the status code specified in the Quality command. Specify the All option to print the actual status code associated with each peak and coincident peak value on the Summary report only. This option is useful for locating edited or possible erroneous data identified by the Load Data Editor and Validation Programs. Specify the No option to indicate that no status codes will be printed.

STOp [**NO** | **YES**]

- **Stop** — Use this command to determine whether the program continues processing if a cut is not found in the database. Specify the No option to allow the Totalizer to continue to process even if the requested cut is not found in the database. Zero values are reported for that customer. Specify the Yes option to stop the Totalizer from processing if a cut is not found in the database.

UOMcheck {**NO** | **YES** [**METhod** | **LEGacy**] } or {**OFF** | **ON**}

- **UOMCheck** — Use this command to determine whether unit-of-measure codes of cuts being accumulated are to be compared. When the Yes or On options are used (or the command is omitted) the unit-of-measure code of each cut in a report block is checked for compatibility with the other cuts in the block (acceptable combinations are listed under the key command).

  When the Yes is used, select the METhod option to allow cuts to be combined if they have the same aggregation or totalization method (Average or Summation). Select the LEGacy option to allow cuts to be combined if they follow the documented UOM combinations (See **UOM Compatibility** on page A-4).

  When the No or Off options are used, this check is bypassed, and cuts are accumulated without regard to their units-of-measure. **Note**: the No or Off option should be used carefully.

  VALid [**NO** | **YES**]

- **Valid** — Use this command to specify whether invalid data should be merged within the reporting period. Specify the No option to indicate that both valid and invalid data may be merged within the reporting period. Specify the Yes option to indicate that only internally valid data may be merged within the reporting period.
- **GRA** — Use this command to produce the SYSGRAPH.HTM file, also in the job folder, which can be used to graph the reported cuts.

Figure 14-2 and Figure 14-3 are sample Environment Files.

```
STATUS      YES
QUALITY     '8'
HIGHEST     4 CPK
VALID       YES
REPORT      SUM
UOMCHECK    NO
```

Figure 14-2 Sample Totalizing Reporter Environment File

```
STATUS      YES
QUALITY     '2'
HIGHEST     3 AVG CPK / * 3 internal peaks only because
VALID       YES         / * AVG and CPK used together
SAVE        REPLACE ARCHIVE
STOP        NO
```

Figure 14-3 Sample Totalizing Reporter Environment File

Step 3: Create Control File (TGX43A.CTL)

The Totalizing Reporter Control File is made up of commands summarized in Figure 14-4. The Control File consists of one or more sets of report blocks. Each report block produces a separate output report. There is no limit to the number of report blocks within a Control File, but there is a *limit of 500 subtotal groups and keys combined within each block*. A report block contains a set of commands that control the report's form and content. A report block is terminated by another BLOCK Command.

BLOck [*block title*]

TLn [*title* | **BLANK** | **NULL**]

DATe *start-time stop-time* [**PAGe**]

AGGregate [*n* | 3600]

SCHedule [*n* | 0]

REMark ['*remark*']

KEY *customer-id channel* [**SUB** | **ADD**] [**MULT** [*n.m* | 1.0]] ['*comment*']

ACCumulate [*subtitle*]

END '*label*' [*customer-id channel*] [**SKIp** *n* | **PAGe**]

Figure 14-4 Format for the Totalizing Reporter Control File

One control command and its associated parameters may appear per line; only the first three letters of each command keyword are required. Blank lines may be used between commands. Blanks must be used as delimiters between parameters. Marginal comments can be included following "/*" on each command line.

Following is a detailed description of each Control File command.

BLOck [*block title*]

- **Block** — Use this command with the TLn command (described below) to indicate the start of a new report. The option "block title" is an optional character string (including blanks) which will appear at the top of each page of the report. If this option is omitted, the top line will include the run date and page number only. There is no limit to the number of report blocks, but there is a limit of 500 subtotal groups and keys combined within each block. A report block is terminated by another Block command.

TLn [*title* | **BLANK** | **NULL**]

- **TLn** — Use this command in addition to the BLOCK, "block title" option to enable up to three additional title lines to be specified. The parameter n must be 2, 3, or 4, indicating the second, third, and fourth title lines, respectively. If the optional "title" parameter on any of these commands is omitted, or the NULL keyword is used, that title line is omitted, and subsequent title lines are printed with no intervening spaces. If the BLANK option is used, a blank line is inserted in the n^{th} position on each report page.

DATe *start-time stop-time* [**PAGe**]

- **Date** — Use this command to establish the reporting period of each report block. The start-time and stop-time are required. Both start-time and stop-time use the format "mm/dd/yy-hh:mm:ss" or "mm/dd/yy" (for whole days). Specify the Page option to start a new page when the date changes on the demand data reports.

AGGregate [*n* | 3600]

- **Aggregate** — Input "n" to establish the number of seconds-per-interval for the data; 60, 300, 900, 1800, and 3600 are permitted values. 3600 is the default.

  Where possible, Oracle Utilities Load Analysis will sum each customer's load data to the specified level. ***Note:*** if the interval length of a customer's data exceeds the value you input, Oracle Utilities Load Analysis will treat that customer as missing.

SCHedule [*n* | 0]

- **Schedule** — Use this command to select a single schedule from the Time-of-Use File (TGE31D) for the time-of-use calculations. The parameter "n" is a non-negative integer

representing a particular schedule in your TOU file. The default, "SCHEDULE 0," automatically defines a single time-of-use period equal to the analysis period. If a schedule number is omitted, do not specify the print option.

> **Note:** Instructions for modifying or adding new schedules to the Time-of-Use File are provided in Step 4. It is very important to keep in mind that any schedule you select must cover the entire analysis period. If the program encounters a gap, it stops processing.

REMark ['*remark*']

- **Remark** — Use this command to print one or more subtitles or comments at the top of the report. The remark field may contain blanks or special characters and must be enclosed in single quotes.

KEY *customer-id channel* [**SUB** | **ADD**] [**MULT** [*n.m* | 1.0]] ['*comment*']

- **Key** — This command is required. The Key command specifies the generic key of each customer cut to be reported; both the customer-id and channel must be supplied. All keys within a report block must have unit-of-measure codes that represent similar data types unless the Environment File UOMcheck command is set to NO or OFF. The following unit-of-measure codes are acceptable combinations:

01 (KWH), 31 (Ind KWH), 32 (KWHr), and 51 (KWH-Out)
02 (KW), and 52 (KW-Out)
03 (KVARH), 33 (Ind KVARH), 34 (KVARH r), 53 (KVARH-Out), 56 (Leading KVARH), 57 (Leading KVARH-Out), 58 (Lagging KVARH), and 59 (Lagging KVARH-Out)
04 (KVAH), 36 (KVAH r)and 54 (KVAH-Out)
05 (TEMP ^{O}F), and 35 (Ind TEMP ^{O}F)
07 (V^2H), and 37 (Ind V^2H)
08 (KQH), 09 (KQH, 45 degrees), 38 (Ind KQH), 39 (KQHr), and 55 (KQH-Out)
11 (Volts), and 41 (Ind Volts)
12 (Amps) and 42 (Ind Amps)
13 (TEMP^{O}C) and 43 (Ind TEMP^{O}C)
17 (MRT) and 47 (Ind MRT)
18 (CMS) and 48 (Ind CMS)
22 (KVAR) and 72 (KVAR-Out)
23 (KVA) and 73 (KVA-Out)

Any other combinations of unit-of-measure codes cause the program to stop processing the current report block and to print out an error message (unless the Environment File command UOMcheck is set to NO or OFF). When such an error occurs, processing continues with the next block of commands. For each key, the user has the option of specifying subtraction (SUB) or addition (ADD). If "SUB" is specified, the channel of data is subtracted from totals or subtotals. If "ADD" is specified, the channel of data is added to totals or subtotals. The default is "ADD". The user also has the option of specifying a multiplication (MULT) factor for an individual key. This option is useful for applying loss factors to data during totalization. Multiplier values may be either positive or negative and may be represented as integer or decimal numbers. A maximum precision of 16 significant digits is allowed. The format of the multiplication parameter is MULT(n.m), where the multiplier value is enclosed in parentheses and no embedded blanks are allowed between the command "MULT" and the left parenthesis. If no multiplier value is included, a default value of "1.0" will be used. A 64-character comment may also be supplied to replace the customer-id on the report. The comment field must be enclosed in single quotes.

ACCumulate [*subtitle*]

- **Accumulate —** Use this command to invoke the totalizing feature of the reporter. Accumulate defines the beginning of a subtotal group; any keys that follow an accumulate command are subtotaled after they are reported. The "subtitle" option is an optional character string (including blanks) which will appear in the left margin before the first line of the subtotal group. ***Note:*** Each ACCUMULATE Command must have a corresponding END Command.

END '*label*' [*customer-id channel*] [**SKIp** *n* | **PAGe**]

- **End —** Use this command to terminate a subtotal group. *There must be one END Command corresponding to each ACCUMULATE Command.* The *required* parameter 'label' is a 64-character string which identifies the subtotal line on the report. It must be enclosed in single quotes and cannot contain blanks. If the subtotal is to be saved, a unique customer-id and channel number must be provided on the END statement. If no customer-id or channel number is given and the SAVE Command is specified in the Environment File, the totaled cut *will not* be saved. The optional SKIP and PAGE parameters provide carriage control after subtotal and totals are printed. "SKIP n" causes "n" blank lines to be left before printing resumes, where "n" is a value from 1 to 9. PAGE causes a page eject before printing resumes.

Figure 14-5 is a Sample Control File.

```
BLO * * * WHOLESALE   DELIVERIES * * *
AGG3600
DAT05/01/98-00:00:00  05/30/98-23:59:59
SCH1 PRINT
REM‘ ACCOUNT NUMBER RECORDER NO.’
REM‘ 45000152800001  ’
REM’45000252800002’
REM‘45000352800010’
REM‘45000452800020’
REM‘45000552800070’
REM‘45000652800100’
REM‘45000752800110’

ACCUMULATE

KEY  N1723 1ADD‘450001’
KEY  N1725 1ADD‘450002’
KEY  N1727 1ADD‘450003’
```

Figure 14-5 Sample Totalizing Reporter Control File

Step 4: Modify the Time-of-Use File (TGY31D.TOU) if Necessary

Refer to **How to Create the Time-Of-Use File (TGY31D.TOU)** in Chapter 5 if you need additional information.

Step 5: Verify the Holiday File (TGY31C.HOL)

Refer to **How to Create the Holiday File (TGY31C.HOL)** in Chapter 5 if you need additional information.

Step 6: Run the Totalizing Reporter Program (X430 or Y450)

After you have created the two input files (Environment and Control Files), and verified the TOU Schedule and Holiday File, you are ready to run the Totalizing Reporter Program. Submit the job using the Totalizing Reporter Submit panel.

The Totalizing Reporter processes each report block separately. Component keys within a block are accessed, aggregated and scanned for data quality. The data is printed in the order specified (subject to intervening subtotals) with up to 12 observations per line.

If the STATUS command has been set to "YES", an asterisk will appear to the right of any data interval whose status code is worse than the quality parameter, or whose subtotal contains a missing data interval. If "STATUS ALL" is specified, the actual status code associated with each peak and coincident peak will be printed on the Summary Report only. If STATUS is set to "NO", the status codes will not be printed, and there is no distinction among normal, non-normal, or missing values.

Each ACCUMULATE command defines the beginning of a new subtotal group. *The command must be followed by a KEY command, END command, or another ACCUMULATE command.* For example, a Control File which specifies:

```
ACC
          KEY       cust-id1
          KEY       cust-id2
             .
             .
             .
          KEY       cust-idn
          END'TOTAL'
```

prints an additional line labeled "TOTAL" after the n-th customer consisting of the column sums of the indicated keys.

ACCUMULATE commands may be nested to form various subtotals. For example, a Control File that specifies:

```
ACC
      ACC
            KEY       cust-idA
            KEY       cust-idB
              .
              .
              .
            KEY       cust-idK
            END'SUBTOTAL 1'
      ACC
            KEY       cust-idL
            KEY       cust-idM
```

```
                    .
                    .
                    .
                KEY      cust-idZ
                END'SUBTOTAL 2'
            END'TOTAL' GRANDTOTAL, 1
```

prints a subtotal line labeled "SUBTOTAL 1" after customer K, another subtotal line labeled "SUBTOTAL 2" after customer Z, followed by a total line labeled "TOTAL". The subtotals consist of the column sums of customers A through K and L through Z, respectively; the total line consists of the column sums of all customers (equivalent to the sum of SUBTOTAL 1 and SUBTOTAL 2). The data labeled "TOTAL" will be saved in the CLDB under the customer-id GRANDTOTAL, in channel 1, using the start date of the report. The label, "TOTAL", will be stored in the descriptor field of the new cut. *The cut will be saved only if the SAVE command is used in the Environment File.*

Please note that the Transformation program (X620) (Y620) may also be used to perform more complex mathematical equations.

Summary Statistics Output File (TGX432) Record Descriptions

The Summary Statistics Report output by the Totalizing Reporter Program (X430/Y450) can be saved to a file in addition to being reported. To produce the file, simply specify a file name after the prompt for Summary Statistics File on the Totalizing Reporter Interface Panel.

The Summary Statistics File consists of two types of records: Block Title Records and Data Records. The Block Title records are identified by a record type of 'B,' and the Data Records are identified by a record type of 'D' for detail (cut) records and 'T' for records representing Subtotals or Totals. The file is delimited to allow easy importation into PC spreadsheet packages. Fields within each record are delimited using the following rules:

- A comma (',') is inserted after each field
- Text fields are enclosed in double quotes (").

Table 1: Block Titles Record

Field Description	Attributes	Comments
Delimiter	Character(1)	
Record Type	Character(1)	'B' = Block Titles Record
Delimiter	Character(3)	
Block Title 1	Character(80)	
Delimiter	Character(3)	
Block Title 2	Character(80)	
Delimiter	Character(3)	
Block Title 3	Character(80)	
Delimiter	Character(3)	
Block Title 1	Character(80)	
Delimiter	Character(1)	
Filler	Character(178)	Blanks

Sample Applications of the Totalizing Reporter

This section includes two sample Totalizing Reporter runs. They are intended to show you how to set up the input files and evaluate the outputs, as well as to give you a broader understanding of how the program's capabilities can be applied in different situations. The examples illustrate totalization of:

- Power Billing
- End Use Study

The samples on the following pages highlight some of the key features of the Totalizing Reporter and may suggest ways in which you can apply the program to your operations.

Here is a list of the two examples and what each is intended to illustrate:

Example A — how to set up the Totalizing Reporter to produce a bill for a customer with multiple locations, and how to provide a summary of each location.

Example B — how to set up the Totalizing Reporter for use with end use studies where it is necessary to compare the total of all end uses to the whole house load to ensure reasonable data is being collected.

For each example, we will look at the input files and the output reports. While each Totalizing Reporter is performed on a group of customers, we will show the output reports for just one representative customer, since the report format is the same for every customer.

Example A — Power Billing

In this example, our objective is to generate a bill for a commercial customer with three locations and also provide a summary of each location. In a situation where several channels of data are recorded for a single customer, a report can be produced to display each channel, subtotals of several channels, and the combined total for each interval of time. The total load can then be saved to the database for future billing reference.

Control File A

```
BLOCK 'ACE INDUSTRIES'
AGGREGATE 3600
DATE 03/01/95-00:00:00 03/31/95-23:59:59
SCHEDULE 1
ACCUMULATE
ACCUMULATE
KEY   N1725  1  ADD  'C110523'
KEY   N1727  1  ADD  'C120523'
KEY   N1736  1  ADD  'C130523'
```

Example B — End Use Study

In this example, data is being collected for an end use survey on household appliances. It is important to sum the end uses to ensure that their total does not exceed the whole house load as a quality control measure. The Totalizer can do this and also show useful statistics such as the contribution of each appliance to total peak load.

Control File B

```
BLOCK ‘COMPARISON OF TOTAL END USE TO

DATE DATE 03/01/95-00:00:00 03/31/95-23:59:59
AGGREGATE 900
ACCUMULATE
KEY C1643HVAJG2  ‘HVAC’
KEY C1643EWHJG3 ADD  ‘ELECWATERHT’
KEY C1643FZRJG4 ADD  ‘FREEZER’
END ‘TOTAL ENDUSE’ TOTC1643 1
ACCUMULATE
KEY C1643TOTJG1  ‘WHOLE-HOUSE’
```

Chapter 15

Key Generator — A Shortcut for Creating Input Files and Reports

This chapter explains the basics of how to use the Key Generator programs. These programs enable you to create lists of cut keys or edit blocks in the database that match user-defined criteria. They are particularly useful for quickly generating Control Files for other programs — such as the Load Data Editor, Cut Series Validation, or Reporter programs.

Topics covered in this chapter are:

- **Introducing the Control Language**
- **Control Language — Logic and Structure**
- **Sample Control Files for Key Generators**
- **Using the Key Generator Programs (X810 or X820)**

With the Key Generator programs, you can create lists of cut keys, edit blocks, or Output Interface files such as CSV, in the database that match specific criteria. For example, you can ask for all customers that have a particular identifier code, or all cuts that have a forced merge field, or all cuts that have a start-date after August 1, 1999. The program is very flexible and the specific criteria are up to you.

You can request that the lists be output as reports or files. The files can be used as input (control) files for other programs — such as the Load Data Editor, Cut Series Validation, or Reporter programs. This means that you can quickly create Control Files according to precise requirements — avoiding the tedious and time-consuming process of manually identifying and typing in each key. The list can consist of full keys, partial keys, or edit blocks.

We will first take a close look at the Control Language that you will use to construct your data requests and format the output. Then we will go through the steps for actually using each program. However any control file may be used for any database.

Introducing the Control Language

The Key Generator Program requires one input file: the Key Generator Control File. You create this file using the "Control Language." You use this Control Language (Figure 15-1) to tell Oracle Utilities Load Analysis how to construct the list of desired keys or edit blocks. Essentially, you specify the logic that Oracle Utilities Load Analysis will use to evaluate each record, or cut, in the database to determine whether or not it meets your criteria.

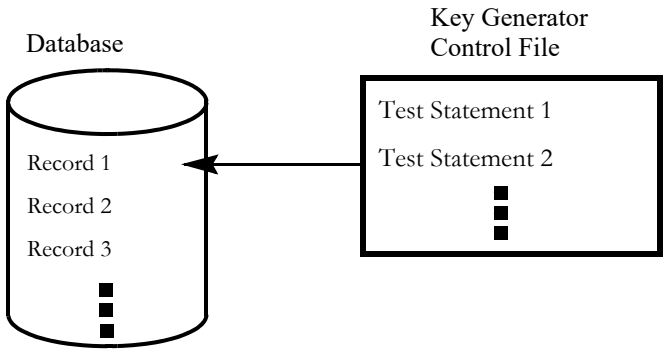


Figure 15-1 List of Test Statements

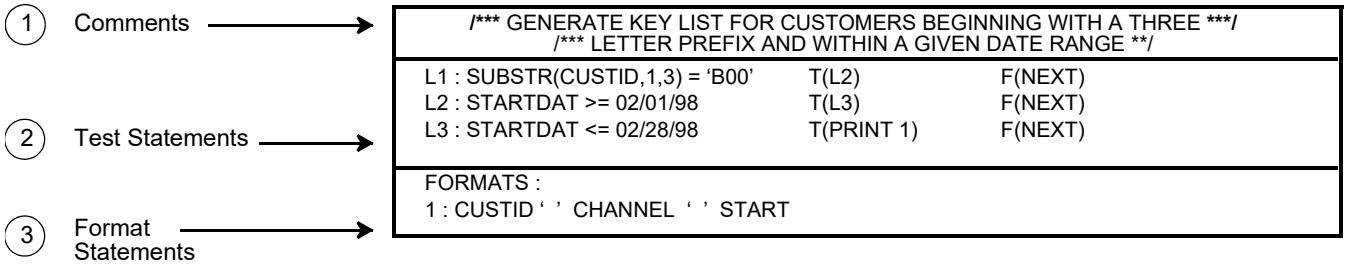
Using the Control Language, you construct a list of test statements that establish your criteria for desired records.

Oracle Utilities Load Analysis applies each test statement to the first record to determine whether or not it meets your criteria. After testing the first record, Oracle Utilities Load Analysis applies the set of test statements to the second record, then the third, and so on until the program has examined the entire database file, or it encounters a STOP Command.

Control Language — Logic and Structure

To learn more about the variables used in the key generator and their formats, see *Appendix C: Key Generator Variable Lists*.

As shown in the example below, the Control Language consists of three major elements:



Test statements **must** precede format statements. Comments can be placed anywhere in the file. Let's look briefly at each element:

1. **Comments** — optional notes that do not affect processing. You can use them to record the purpose of the file, for example.

   > **Note:** You must precede your comments with the symbol: /*. Do not split comments across lines, unless you begin and end each line with the slash/ asterisk symbols.

2. **Test Statements** — compare a single field in each record (called the "variable") to your criteria (called the "test-value"), and indicate what action should be taken depending upon the results of the comparison.

3. **Format Statements** — specify how the results should be organized in the report or output file (called the "Extracted Data File"). You can specify what information you want printed (including data from any fields in the selected records, as well as titles and notes) and how you want the information organized in the file or on the page. Format statements are typically activated by PRINT instructions in test statements (Figure 15-2).

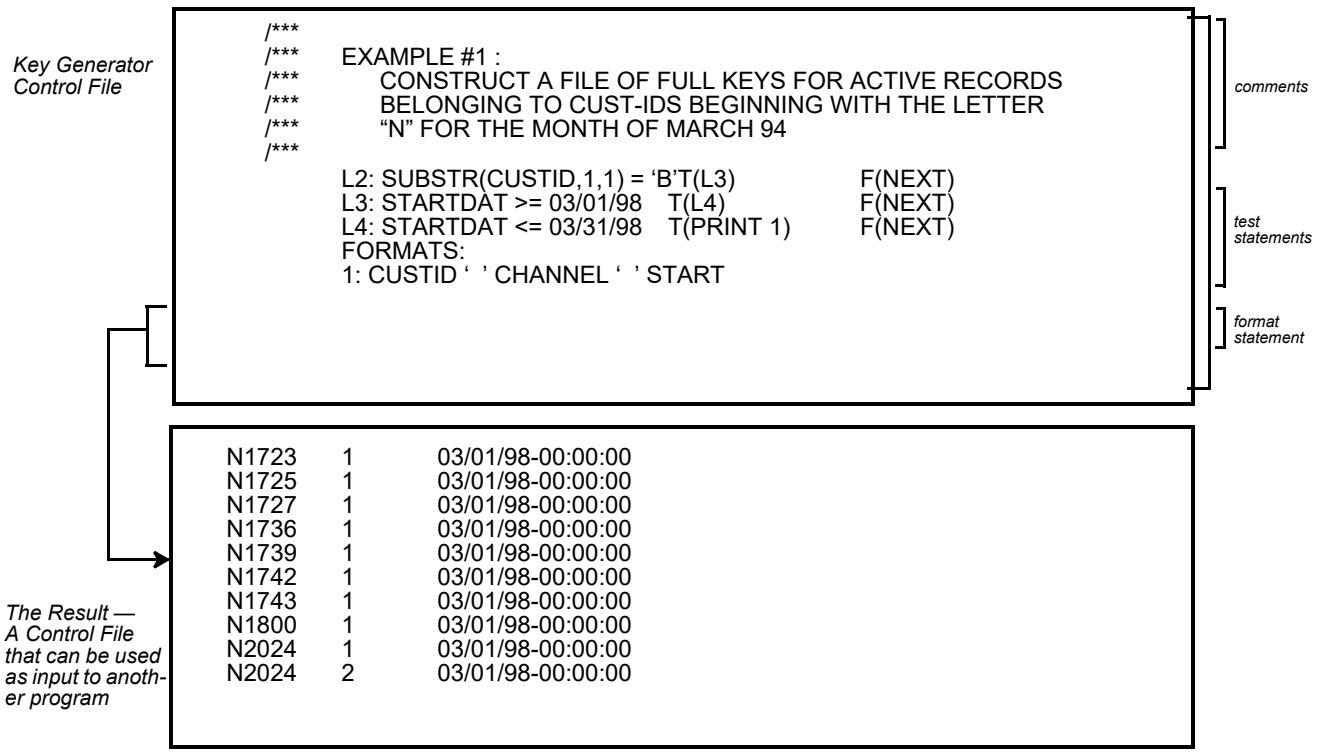


Figure 15-2 Sample Key Generator Control File and Its Results

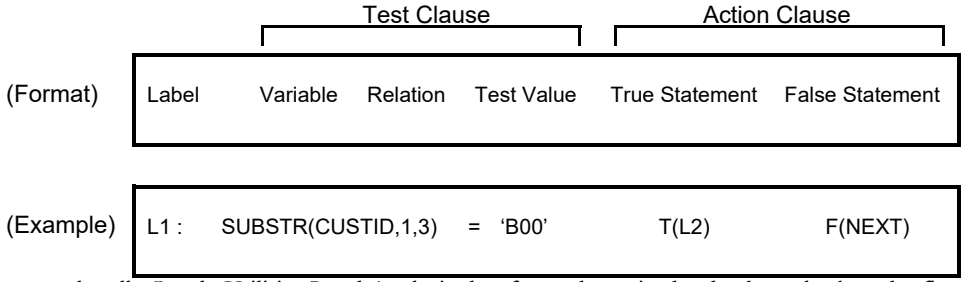
The example shows how PRINT instructions trigger format statements, which, in turn, dictate the contents and organization of the output.

Counters are also important. They are special elements that you can use in different ways throughout the query — for example, to count the number of records with specified characteristics, or to stop processing after a certain number of records have been counted.

Test statements, format statements, end statements, and counters can be complex, so we will examine each of them more closely in the following sections.

Test Statements

You can input up to 2000 test statements in a single Control File, but only one test statement per line. Each test statement consists of the following elements:



The example tells Oracle Utilities Load Analysis that for each cut in the database, look at the first three characters in the customer-id. If it is true that they match B00, then compare that cut against the criteria specified by test statement L2. If it is false, skip that record and test the next cut in the database or file.

To better understand how you can construct your own test statements, let's look at each generic element of the format:

- **Label** — Identifies the test statement for reference by other statements. (Labels are optional; you need to preface a statement with a label only if it will be the target of other statements.) A label can be any word or code of your own choosing, as long as it is no more than 8 alphanumeric characters long, and the first character is alphabetic. Input a colon (:) after the label, but do **not** insert a blank between the label and the colon. Do insert a blank after the colon and before the test clause. Here are three sample labels:

  L1:
  LINE:
  TSTCHANL:

- **Variable** — The record field being examined. The acceptable variables and their formats for Key Generators are listed in tables in *Appendix C: Key Generator Variable Lists* and summarized in the *Oracle Utilities Load Analysis Quick Reference Guide.*

  Note: *The variables you specify for Key Generators must be one of these fields. Options are slightly different depending upon which database you are working with, so be sure to check the appropriate table for the database.*

  Whenever a test variable is defined as "CHAR" in the variable list, for example, CUSTID = 'A7304', and your test value for that variable is a text string, you must enclose the test value in single quotes.

- **Relation** — How the record value should compare to the test-value(s). You can use any of the following relations. "Equal to" is the default.

  = variable equal to test-value
  >greater than
  <less than
  # or ^= not equal to
  >=greater than or equal to
  <= less than or equal to
  =*variable contains test-value anywhere
  =?variable contains test-value starting in a specific column

- **Test value** — Criteria for the evaluation. It can be a constant or character string that may exist in a database record, or it can be a second variable. For example, consider the following test clauses:

  ```
  DESC     =  'NONE'
  DESC     =  CUSTID
  ```

  The first sample test statement compares a variable to a constant; the second compares a variable to a second variable.

  You can include multiple test-values within a single test statement. In such cases, a logical OR is assumed. For example, *all* of the following test clauses mean "Is the variable CUSTID equal to A7304 or B7418 or C7313?".

  ```
  CUSTID   =  'A7304'  =  'B7418'  =  'C7313'
  CUSTID      'A7304'     'B7418'     'C7313'
  CUSTID   =  'A7304',    'B7418',    'C7313'
  ```

  If you want a variable to meet multiple conditions, you must put each test value on a separate line. For example, if you wanted all customer records with strata numbers greater than or equal to 2 **and** less than or equal to 4, you could construct the following test statements:

  ```
  CHANNEL  >= 2
  CHANNEL  <= 4
  ```

  If a test value contains embedded blanks or commas, you must enclose it in single quotes.

Before we go on, let's look at a few more sample test clauses and their meanings (test clauses are a subset of the test statement and consist of a variable, relation, and one or more test values).

CHANNEL # 1	Is the variable CHANNEL not equal to 1?
CUSTID = 'RAEH', 'RHWH'	Is the variable CUSTID equal to the constant RAEH or RHWH?
DESC = CUSTID, 'TEMP05'	Is the variable DESC equal to the variable CUSTID or the constant TEMP05?

- **Wild cards**

A "wild card" feature allows you to find particular characters within a variable. The "=*" operator enables you to find the test value anywhere in the variable. For example, the following test clause means "Does the variable CUSTID Contain AGG?".

```
CUSTID    =*      'AGG'
```

Another option, the "=?" operator, allows you to find the test value starting in a specific column. To use this positional wildcard operator, the test value is specified with placeholders ("?") preceding the starting column. For example, the following test clause means "Does the variable CUSTID contain AGG starting in column 4?".

```
CUSTID    =?      '???AGG'
```

The wildcard feature can only be used on database variables that are defined as character. See the variable lists in *Appendix C: Key Generator Variable Lists*.

- **True and False clauses** — True clauses specify the action the program will take if a record matches the criteria. False clauses specify the action the program will take if a record does not match the criteria. Possible actions are:
  - Output one or more fields from the record (PRINT)
  - Count occurrence (COUNT)
  - Continue with the next test (continue)
  - Restart the test with the next record (NEXT)
  - Branch forward to another test statement (label)
  - Terminate the program (STOP)
  - A combination of the above.

Here is a summary of the format you will use for constructing the clauses. ***Note:*** True clauses must be preceded by the prefix "T" and false clauses by "F". Parentheses must enclose the clauses. Do not insert a blank between the prefix and the left parenthesis.

True Clause

T ([**PRINT** *n*] [,**COUNT** *m*] [,*label* | ,**STOP** | ,**NEXT** | , continue])

False Clause

F ([**PRINT** *n*] [,**COUNT** *m*] [,*label* | ,**STOP** | ,**NEXT**])

PRINT n — tells Oracle Utilities Load Analysis to write data from the record to a file or report, using the format you specify in format statement "n". (Input the word "PRINT" followed by a blank and the format statement number "n".)

COUNTm —tells Oracle Utilities Load Analysis to count the number of occurrences of the condition you specified in the test statement. You can have multiple counters in a Control File; use an integer "m" to distinguish between them. (Input the word "COUNT" followed

by an integer "m".) *Note:* For a more complete description of what you can do with COUNTm, see *Counter Variables* on page 15-11. "m" can be a value between 1 and 999.

NEXT — tells Oracle Utilities Load Analysis to stop evaluating the current record and compare the next record with the first test statement. This action is the default for all false clauses and the last true clause in the file. (For true clauses, input the word "NEXT"; for false clauses, no input is required.)

label — tells Oracle Utilities Load Analysis to compare the current record with the test statement prefixed by "label". (Input your designation for the label. *Note:* The labelled statement must occur below the clause; in other words, only "forward" branches are allowed.)

STOP — tells Oracle Utilities Load Analysis to stop the program whenever the condition specified has been met. STOP is useful when searching for one particular record; there is no need to scan the remaining records after finding the desired record. (Input the word "STOP".)

continue— tells Oracle Utilities Load Analysis to compare the current record with the next test statement. This action is possible only for true clauses and is the default for all true clauses except the last one in the file. (Because it is the default, no entry is required. Do not input the word "continue".) To continue when a false clause is executed, you must input the label of the next statement.

Logically, each test statement must have one true clause and one false clause. However, in many cases you will not need to actually input both, because you can often rely on the defaults.

Oracle Utilities Load Analysis will compare each record against the entire set of test statements until it reaches a STOP statement or the end of the database. It will then output the results according to the instructions you provide in the format statements.

Comparison Test Statements for Intervals and Statuses

Comparison test statements allow you to specify test statements that compare interval values and status codes to values specified in the test statement.

The format for these statements is as follows:

<test_variable> (<time-range>) <operator> <value>

where:

- <test_variable> is the variable used in the test statement. There are two supported variables for comparison test statements: INTERVAL and STATUS.
    - INTERVAL Returns TRUE for each cut that contains interval values that match the specified value.
    - STATUS Returns TRUE if status code values with each cut match the specified value.
- <date-range> is the date range that a cut's intervals or statuses must fall within to be included in the test. There are two formats for the date range:
    - (*) specifies no date range. All intervals with each cut are tested.
    - (<start-time> [<stop-time>]) defines a specific date range. Both date (09/26/06) or date/time (09/26/06 00:00:00) format can be used to specify the dates.
        - <start-time> if provided, all intervals beginning at the specified time index are tested.
        - <stop-time> if provided, all intervals within the specified time range are tested. Cuts that do not fall within the specified date range fail the test.

- <operator> is the comparison operator for the test. Available operators include:

Symbol	Meaning
=	Returns TRUE if all intervals/status codes are equal to the test value.
>	Returns TRUE if all intervals/status codes are greater than the test value.
<	Returns TRUE if all intervals/status codes are less than the test value.
# or ^=	Returns TRUE if all intervals/status codes are NOT equal the test value.
>=	Returns TRUE if all intervals/status codes are greater than or equal to the test value.
<=	Returns TRUE if all intervals/status codes are less than or equal to the test value.
*=	Returns TRUE if the cut contains intervals/status codes equal to the test value.
*>	Returns TRUE if the cut contains intervals/status codes greater than the test value.
*<	Returns TRUE if the cut contains intervals/status codes less than the test value.
*>=	Returns TRUE if the cut contains intervals/status codes greater than or equal to the test value.
*<=	Returns TRUE if the cut contains intervals/status codes less than or equal to the test value.

- <value> is the value to which the intervals are compared. Values must be within single quotes (‘ ’).

INTERVAL

The INTERVAL test returns TRUE for each cut that contains interval values that match the specified value.

Example 1: Returns true for all cuts that contain an interval value of 0:

```
L2: INTERVAL(*) *= 0 T(PRINT) F(NEXT)
```

Example 2: Returns true for all cuts in which all interval values within the cut are greater than or equal to 0 from 8/1/2006 and beyond:

```
L2: INTERVAL(08/01/06) >= 0 T(PRINT) F(NEXT)
```

Example 3: Returns true for all cuts that contain an interval value of 0 within the month of August 2006:

```
L2: INTERVAL(08/01/06 08/31/06) *= 0 T(PRINT) F(NEXT)
```

The resolution provided as the test value will be the resolution in which test data will be scaled up to for comparison. For example, if a value of 100 with no decimal places is provided, the program will round each and every interval up to whole numbers (no decimal digits) before being compared to the value of 100. Alternatively, if a value of 100.0 is provided, then all interval values will be rounded up to 1 decimal place before being compared to the value (100.0).

Note: The COUNTINT counter variable can be used with INTERVAL tests to count the number of intervals that pass the test. See **COUNTINT** on page 15-13 for more information.

STATUS

STATUS Returns TRUE if status code values with each cut match the specified value.

Example 1: Returns true for all cuts that contain status code '2':

```
L2: STATUS(*) *= '2' T(PRINT) F(NEXT)
```

Example 2: Returns true for all cuts in which status codes within the cut are greater than or equal to '2' from 8/1/2006 and beyond:

```
L2: STATUS(08/01/06) >= '2' T(PRINT) F(NEXT)
```

Example 3: Returns true for all cuts that contain status code '2' within the month of August 2006:

```
L2: STATUS(08/01/06 08/31/06) *= '2' T(PRINT) F(NEXT)
```

When greater than or less than comparisons are applied to status codes, the comparison logic operates as follows. Better status codes are considered as having lower value than the worse status code. Thus a status code of '9' has the highest value and a blank status code (' ') has the lowest. For example, status code 'A' < 'B' and status code '1' < '9', etc.

> **Note**: The COUNTSTAT counter variable can be used with STATUS tests to count the number of intervals that pass the test. See **COUNTSTAT** on page 15-13 for more information.

Some Tips on Constructing Test Statements

Before going on to the format statements, let's look at a few sample test statements. These samples show that there can be many ways of setting up the test statements to accomplish the same task, but you must be careful to structure the statements to get the result you really want. In addition, some approaches are more direct and therefore easier to understand. As a general rule, it is best to construct the file with the fewest possible statements, and to set the clauses to match desired conditions. The language is designed so that the default for each true clause is continue testing and the default for each false clause is to get the next record and start the tests again. Therefore, if you set up each test so that the result you desire is true, then you can take advantage of the defaults and simplify each statement. Sample 1 below is equivalent to Sample 2, but much simpler and easy to understand.

In the following examples, our task is to direct the program to print data from all records that have strata numbers 2 through 10. The first two statements will yield the proper results; the third will not.

Sample 1

```
CHANNEL >= 2
CHANNEL <= 10    T(PRINT 1)
```

The first test statement asks, "Is the CHANNEL greater than or equal to 2?" Because we have not supplied any action clauses, Load Analysis will assume the defaults. If true, it will compare the current record to the next test statement. If false, it will skip to the next record. The second test statement asks, "Is the stratum less than or equal to 10?" If true, Load Analysis will output data according to format statement 1. If false, it will skip to the next record. This is a logical and efficient approach.

Sample 2

```
             CHANNEL < 2 T(NEXT), F(TRYAGAIN)
TRYAGAIN:    CHANNEL   > 10  T(NEXT), F(PRINT 1)
```

The first test statement asks, "Is the CHANNEL less than or equal to 2?" If yes, the program will skip the current record and will continue testing with the next record. If false, it will branch to the test statement labelled "TRYAGAIN." The "TRYAGAIN" statement asks, "Is the CHANNEL greater than or equal to 10?" If yes, Load Analysis will skip the record, but if not, it will output data according to format statement 1.

Sample 3 (incorrect)

```
CHANNEL >= 2, <= 10 T(PRINT 1)
```

While this approach will yield the desired result, it is not as direct and efficient as Sample 1.

*This is an incorrect approach. Because Load Analysis always assumes you mean "OR" between multiple test-values, this statement reads, "If the CHANNEL is greater than or equal to 2, **or** less than or equal to 10, output data from the record according to format statement*

Format Statements

You use format statements to specify the layout of the output file or report. You can tell the program to print variables from the cuts or file records, add titles or notes, and insert blank lines or spaces to make the output more readable. Remember, Oracle Utilities Load Analysis executes a format statement only when directed by the 'PRINT n' option of a true or false clause. (The exception is END statements, which are explained at the end of this section.)

Let's consider an example:

```
CHANNEL = 2, 3, 4 T(PRINT 1)
FORMATS:
1: 'CUSTOMER IS' CUSTID 'AND STRATUM IS' CHANNEL SKIP(1)
```

For each customer whose strata number is 2, 3, or 4, Oracle Utilities Load Analysis will output a line in the report or file using the specified format. For example, for customer BOO3 who is in stratum 3, Oracle Utilities Load Analysis will output the following line, preceded by a blank line:

CUSTOMER IS BOO3 AND CHANNEL IS 3

You can have up to 2000 format statements in a single Control File, and each one can continue over more than one line. Within the file, you must put the format statements *after all* of the test statements (you cannot intermingle the two types). You must input the Keyword "FORMATS:" between the test statements and the format statements.

Each format statement has the general form shown below. You can input the parameters in any order in the output line, except "n" must begin each line and "field spec" (if used) must come after the variable it refers to.

n: {[*variable*[*field spec*]] [*literal*] [**BLANK**(*s*)] [**PAGE**] [**SKIP**(*y*)]

- **n:** — identifies the format statement for reference by test statements. It can be any integer from 1 to 99; you need not enter them consecutively. The integer "n" must be followed by a colon (:). Do not put blanks between "n" and the colon.
- **variable** — identifies a variable in the record, which you want written to the output report or file. You can use any name from the variable list in *Appendix C: Key Generator Variable Lists* or the *Oracle Utilities Load Analysis Quick Reference Guide*. You can have any number of variables in a format statement.
- **field-spec** — You may choose from two forms of this optional parameter.

  The first is designed for use following a **numeric variable** and is of the form:

  w[.#dec]

  where **w** represents the total length of the output field, and the optional **#dec** preceded by a decimal point represents the number of decimal digits to be printed. For example, to specify an output field width of 7 characters including 2 decimal digits, you would code "7.2"; to specify the same width with no decimal digits, you would code "7". If you specify a field width less than the length of the variable's value, Oracle Utilities Load Analysis outputs only the number of characters specified. If you specify a width greater than the numeric variable's length, the output field is padded on the left with blanks to the width specified.

  The second form of the field spec may be used ***only*** following a **character variable** and is of the form:

  startpos/[endpos]

  where **startpos** is a positive integer representing the first position within the variable that you want to output, and is followed by a slash (/), which is *required* to identify this form. If the **startpos** specified is greater than the length of the variable, the format statement will be rejected. After the slash you may code the last position within the variable that you want written (**endpos**), which must be a positive integer greater than or equal to the start position, and less than or equal to the length of the variable. The **endpos** parameter is optional; if you omit it, Oracle Utilities Load Analysis will write out all characters from the indicated start

position through the end of the variable. For example, to write out positions 11 - 20 of an 80-character variable, you would code "11/20". To write out all characters of the variable starting from position 11 (i.e., positions 11 - 80), you could code either "11/80" or "11/".

> **Note:** If you use the other form of the field spec, containing no slash, after a character variable, the first number is considered a width rather than a start position, and Oracle Utilities Load Analysis will write out the first **w** characters of the variable and will ignore any decimal point and second number. If the specified width is greater than the character variable's length, the output field will be padded on the right with blanks to the width specified.

- **literal** — an optional character string, not exceeding 80 characters. Use it to input titles or notes, or to insert spaces between variables in the report or file. You **must** enclose your entry in single quotation marks. If your character string includes single quotation marks, they must be replaced with the escape character """. If you wish to print out a comment beginning with "/*", you must prefix comment characters with an ampersand (&).

  Example: The following command:

  ```
  FORMATS:
  1: '&/*' TRIM(CUSTID) ' ' TRIM(CHANNELL
  ```

  produces the following:

  ```
  /* N1723 1
  ```

- **BLANK(s)** — use to specify that "s" blank spaces be inserted in the output line. You can specify from 1 to 132 spaces. You must enclose the number in parentheses.
- **PAGE** — use to force a page eject before the output line.
- **SKIP(y)** — use to specify "y" blank lines before the output line. You must enclose the number in parentheses.

TRIM [*format-variable* | *literal*

- **TRIM [format-variable|literal]** — use to trim leading and trailing spaces from any output variable or literal string.

END: {[*variable*[*field spec*]] [*literal*] [**BLANK**(*n*)] [**PAGE**] [**SKIP**(*n*)] [**TRIM** (*format-variable* | *literal*)]}

- **END Statement** — The END Statement is a special type of optional format statement, which is not tied to a PRINT instruction in the test statements. All other format statements are executed while the database is being processed and PRINT actions are encountered. The END Statement is only executed when a STOP action occurs or the end of the database is reached. It is normally used to print out summary information and counter values. An example of an END Statement is:

  END: "NUMBER OF COMMERCIAL APARTMENT BUILDINGS IS " COUNT1

  END statements use the same format as regular format statements, except you input the keyword "END" instead of a variable "n".

  > **Important Note:** The keyword "END" must be used as a component of a format statement. It is not used to indicate the end of a file. This will cause the program to abort.

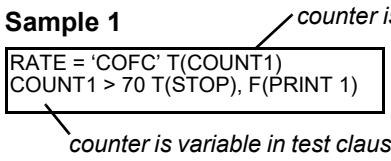
Counter Variables

Counter variables enable you to keep a running count of the occurrence of specified conditions as the program applies your test statements to the records. They can be used in several different ways to accomplish different tasks:

- *as the specified action in true/false clauses* — to count the number of records matching or not matching your criteria.
- *as the variable in test clauses* — to initiate other actions when a certain count has been reached.
- *as a variable in format statements* — to output the number counted. Counter variables are treated as integer variables with a default field width of 8.

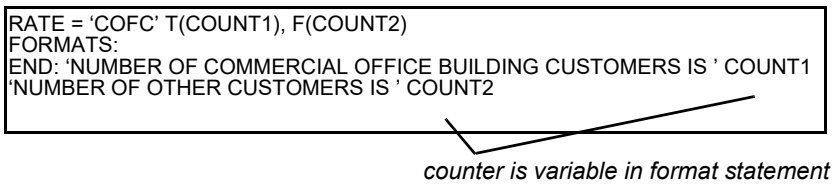
The following two examples illustrate the three different applications:

Sample 1



These two test statements will cause data for the first 70 COFC (Commercial Office Building) customers to be printed out.

Sample 2



The number of each type of customer will appear at the end of the report.

Remember, the format of counters is "COUNTm", with the integer "m" used to distinguish the different counters. You can have up to 999 counters in a Control File.

COUNTINT

COUNTINT is a counter variable that can be used in TRUE and/or FALSE statements to store a count of the number of intervals that pass the INTERVAL test function. See **Comparison Test Statements for Intervals and Statuses** on page 15-7 for more information about the INTERVAL test function. Once stored, COUNTINT can be used like other counter variables for further testing logic.

If you include multiple INTERVAL tests in your key generator file, you can use the COUNTINT++ counter variable in place of COUNTINT to keep a running total of all the intervals that pass any of the INTERVAL test functions.

COUNTSTAT

COUNTSTAT is a counter variable that can be used in TRUE and/or FALSE statements to store a count of the number of intervals that pass the STATUS test function. See **Comparison Test Statements for Intervals and Statuses** on page 15-7 for more information about the STATUS test function. Once stored, COUNTSTAT can be used like other counter variables for further testing logic.

If you include multiple STATUS tests in your key generator file, you can use the COUNTSTAT++ counter variable in place of COUNTSTAT to keep a running total of all the intervals that pass any of the STATUS test functions.

SUBSTR Functions

The SUBSTSR function extracts a section of characters from a specified variable. This function can be applied to character variables in both test statements and format statements. The SUBSTR function has the following format:

```
SUBSTR(variable, start-position, length)
```

variable: The Key Generator variable on which to operate. The variable must be a character datatype.

start-position: The location in the variable where the function begins extracting characters. The first character of the variable has a start-position of 1.

length: The desired number of characters to extract. Specify "*" to extract to the end of the variable.

The following table shows the results of several SUBSTR functions applied to a DESC variable with the value "250 BROADWAY LANGLEY NC 07034":

Function	Result
SUBSTR(DESC, 1, 3)	"250"
SUBSTR(DESC, 14, *)	"LANGLEY NC 07034"

Control File Examples:

Test Statement: L1: SUBSTR(CUSTID,1,1) = 'N' T(PRINT 1) F(NEXT)

Format Statement: 1: SUBSTR(CUSTID,1,10) ',' SUBSTR(DESC,1,10)

Sample Control Files for Key Generators

To further illustrate how you can apply the Control Language we just described, here are some sample Key Generator Control files.

Example #1 — CLDB Key Generator Control File

This Control File instructs the CLDB Key Generator Program to generate a list of all active records whose customer-id begins with either "B0" or "B1" and whose district numbers fall within the specified criteria.

```
/*** GENERATE A KEY LIST FOR ALL CUTS WITH ***/
/*** A SPECIFIC GEOGRAPHICAL DISTRICT NUMBER ***/

L2:     SUBSTR(CUSTID,1,2)='B0'          T(L4)        F(L3)
L3:     SUBSTR(CUSTID,1,2='B1'           T(L4)        F(NEXT)
L4:     SUBSTR(DESC,1,3)>='700'          T(L5)        F(L6)
L5:     SUBSTR(DESC,1,3)<='900'          T(PRINT 1)   F(NEXT)
L6:     SUBSTR(DESC,1,3)>='000'          T(L7)        F(NEXT)
L7:     SUBSTR(DESC,1,3)<='099'          T(PRINT 2)   F(L8)
L8:     SUBSTR(DESC,1,3)>='300'          T(L9)        F(L10)
L9:     SUBSTR(DESC,1,3)<='499'          T(PRINT 2)   F(NEXT)
L10:    SUBSTR(DESC,1,3)>='100'          T(L11)       F(L12)
L11:    SUBSTR(DESC,1,3)<='199'          T(PRINT 3)   F(L12)
L12:    SUBSTR(DESC,1,3)>='500'          T(L13)       F(NEXT)
L13:    SUBSTR(DESC,1,3)<='699'          T(PRINT 3)   F(NEXT)


FORMATS:
1:      CUSTID      '   '     CHANNEL
2:      CUSTID      '   '     CHANNEL
3:      CUSTID      '   '     CHANNEL
```

Example #2 — CLDB Key Generator Control File

This Control File tells the CLDB Key Generator Program to construct edit blocks for the two cut series "B021, 1" and "B037, 1". The resulting file, shown below, can in turn be used as input to the Load Data Editor Program, to actually apply the edits to the cut series.

```
/***
/*** KEY GENERATOR:
/*** EDITOR EXAMPLE. GENERATE EDIT BLOCKS FOR
/*** CUTS 'B021' AND 'B037'
/***


          SUBSTR(CUSTID,1,5)='B021'
          CHANNEL    =        1                                  F(CHECK2)
                                                    T(PRINT 1)
CHECK2:   SUBSTR(CUSTID,1,5)='B037'
          CHANNEL    =        1
                                                    T(PRINT 2),  F(PRINT 3)



FORMATS:
1:   'KEY '      CUSTID'   '       CHANNEL    '   '   START    SKIP(1)
'        '   'SET METER_MULT 4.02'                              SKIP(1)
'        '   'SET METER_OFFSET 2.0'                             SKIP(2)

2:   'KEY '      CUSTID'   '       CHANNEL    '   '   START    SKIP(1)
'        '   'SET UOM 01'                                       SKIP(2)

3:   'KEY '      CUSTID'   '       CHANNEL    '   '   START    SKIP(1)
'        '   'SET UOM 03 '                                      SKIP(2)
```

Example 2 — Output File

```
KEY      B021                          1       07/01/98-00:00:00
         SET METER_MULT 4.02
         SET METER_OFFSET 2.0
KEY      B037                          1       07/01/98-00:00:00
         SET UOM 01
```

Example #3 — CLDB Key Generator Control File

Example 3 shows a Control File that generates edit blocks for customer-ids starting with R2921 and N2024. For channel 1 of customer-ids starting with R2921, an edit block is written consisting of a KEY Command and two correction commands, "SET METER-MULT 4.02" and "SET MERGE YES". For channel 1 of customer-ids starting with N2024, edit blocks containing a KEY Command and a "SET ARCHIVE YES" correction command are generated. All other customer-ids starting with N2024 have edit blocks that contain commands to set their merge flag to "YES". Output from this example is shown in the lower box.

```
      SUBSTR(CUSTID,1,5) = 'R2921'F(CHECK2)
      CHANNEL = 1        T(PRINT 1)
CHECK2:SUBSTR(CUSTID,1,5) = 'N2024'
      CHANNEL = 1          T(PRINT 2), F(PRINT 3)

FORMATS:
1:    'KEY ' CUSTID ' '  CHANNEL '  '     START       SKIP(1)
'   ' 'SET METER_MULT 4.02'                           SKIP(1)
'   ' 'SET MERGE YES'                                 SKIP(2)
2:    'KEY ' CUSTID  '  '  CHANNEL '  '   START       SKIP(1)
'   ' 'SET ARCHIVE YES'                               SKIP(2)
3:    'KEY ' CUSTID  '  '  CHANNEL '  '   START       SKIP(1)
'   ' 'SET MERGE YES'                                 SKIP(2)
```

Control File

```
          KEY N2024   1  05/14/99-09:55:00 SET ARCHIVE YES

          KEY N2024   1  06/14/99-14:27:00 SET ARCHIVE YES

          KEY N2024     205/14/99-09:55:00 SET MERGE YES

          KEY N2024   2  06/14/99-14:27:00 SET MERGE YES

          KEY R2921   1  04/12/99-10:10:00 SET METER_MULT 4.02SET MERGE YES

          KEY R2921   1  05/10/99-10:27:00 SET METER_MULT 4.02SET MERGE YES

          KEY R2921   1  06/09/99-23:15:00 SET METER_MULT 4.02SET MERGE YES
```

Output File

Example #4 — CLDB Key Generator Control File

Example 4 shows a Control File that generates full keys from cuts within the month of March 1998 for customer-ids beginning with the letter 'B'.

```
L2:   SUBSTR(CUSTID,1,1)= 'B'       T(L3)        F(NEXT)
L3:   STARTDAT    >= 03/01/98       T(L4)        F(NEXT)
L4:   STARTDAT    <= 03/31/98       T(PRINT 1)   F(NEXT)

FORMATS:
```

Example #5 — CLDB Key Generator Control File

Example 5 shows a Control File that generates full keys of invalid cuts starting after the date 01/01/98. Because the last cut of a cut series is always marked as externally invalid, it will be excluded from this report *unless* it is internally invalid.

```
/*     IF INTERNALLY VALID THEN GOTO L2.              */
/*     ELSE PRINT IF AFTER 01/01/98 AND               */
/*     START TESTING WITH NEXT RECORD                 */
       L1: INTVALID = NOT(L14)F(L2)
/*  IF EXTERNALLY INVALID THEN GOTO L3.               */
/*  ELSE START TESTING WITH THE NEXT RECORD           */

    L2:   EXTVALID = YES       T(NEXT)           F(L3)
    L3:   MSG1 = '  '          T(NEXT)           F(L4)
    L4:   MSG1 > '(E'          T(L14)            F(L5)
    L5:   MSG2 > '(E'          T(L14)            F(L6)
    L6:   MSG3 > '(E'          T(L14)            F(L7)
    L7:   MSG4 > '(E'          T(L14)            F(L8)
    L8:   MSG5 > '(E'          T(L14)            F(L9)
    L9:   MSG6 > '(E'          T(L14)            F(L10)
    L10:  MSG7 > '(E'          T(L14)            F(L11)
    L11:  MSG8 > '(E'          T(L14)            F(L12)
    L12:  MSG9 > '(E'          T(L14)            F(L13)
    L13:  MSG10 > '(E'         T(L14)            F(NEXT)
    L14:  STARTDAT > 01/01/98  T(PRINT 1, NEXT)  F(NEXT)
    FORMATS: /** FORMAT STMTS GO BELOW TEST STMTS. **/
```

Example #6 — CLDB Key Generator Control File

Example 6 shows a Control File that generates edit blocks for all cuts that are invalid. Because the last cut of a cut series is always externally invalid, it is not included in this report.

```
/*     IF INTERNALLY VALID THEN GOTO L2. ELSE PRINT AND      */
/*     START TESTING WITH NEXT RECORD                        */
   L1: INTVALID = NO    T(PRINT 1, NEXT) F(L2)
/*     IF EXTERNALLY INVALID THEN GOTO L3. ELSE START        */
/*     TESTING WITH THE NEXT RECORD                          */

L2:   EXTVALID = YES       T(NEXT)              F(L3)
L3:   MSG1 = '  '          T(NEXT)              F(L4)
L4:   MSG1 > '(E'          T(L14)               F(L5)
L5:   MSG2 > '(E'          T(L14)               F(L6)
L6:   MSG3 > '(E'          T(L14)               F(L7)
L7:   MSG4 > '(E'          T(L14)               F(L8)
L8:   MSG5 > '(E'          T(L14)               F(L9)
L9:   MSG6 > '(E'          T(L14)               F(L10)
L10:  MSG7 > '(E'          T(L14)               F(L11)
L11:  MSG8 > '(E'          T(L14)               F(L12)
L12:  MSG9 > '(E'          T(L14)               F(L13)
L13:  MSG10 > '(E'         T(PRINT 1, NEXT)     F(NEXT)
L14:  STARTDAT > 01/01/98  T(PRINT 1, NEXT)     F(NEXT)
FORMATS: /** FORMAT STMTS GO BELOW TEST STMTS.              **/
1:    'KEY' CUSTID '  ' CHANNEL '  ' START
```

Example #7 — ALDB Key Generator Control File

This example shows a Control File that generates full keys of all cuts with customer-ids starting with "ACC2" that start in the month of February 1998.

```
/*** GENERATE KEY LIST FOR CUSTOMERS BEGINNING WITH A    **/
/*** FOUR LETTER PREFIX AND WITHIN A GIVEN DATE RANGE    **/


L1:   SUBSTR(CUSTID,1,4) = 'ACC2'T(L2)    F(NEXT)
L2:   STARTDAT >= 02/01/98 T(L3)          F(NEXT)
L3:   STARTDAT <= 02/28/98 T(PRINT 1)     F(NEXT)
FORMATS: /** FORMAT STMTS GO BELOW TEST STMTS.          **/
1: CUSTID '  ' CHANNEL '  ' START
```

Example #8 — ALDB Key Generator Control File

This example shows a Control File that generates full keys for all cuts whose start-times fall on or after January 1, 1998. This Key File is to be used in a run to segment the ALDB into parts. The outputs from this example are displayed on the next page.

```
START >= 01/01/98-00:00:00 T(PRINT 1)

FORMATS:
1:  TRIM (CUSTID) '   '   TRIM (CHANNEL) '  '    TRIM(START)
```

Control File

```
N1723 1 01/12/98-10:12:00
N1723 1 02/11/98-12:23:00
N1723 1 03/16/98-11:35:00
N1723 1 04/12/98-13:51:00
N1725 1 01/07/98-13:03:00
N1725 1 02/09/98-12:25:00
N1725 1 03/15/98-13:09:00
N1725 1 04/12/98-11:10:00
N1727 1 01/08/98-15:10:00
N1727 1 02/11/98-11:57:00
N1727 1 03/16/98-11:03:00
N1727 1 04/12/98-12:40:00
N1736 1 01/08/98-13:48:00
N1736 1 02/11/98-10:56:00
N1736 1 03/16/98-10:03:00
N1736 1 04/13/98-09:49:00
N1739 1 01/08/98-12:55:00
N1739 1 02/10/98-10:50:00
N1739 1 03/15/98-14:42:00
N1739 1 04/13/98-11:04:00
N1742 1 01/08/98-11:58:00
N1742 1 02/10/98-11:19:00
N1742 1 03/15/98-14:55:00
N1742 1 04/13/98-11:18:00
N1743 1 01/08/98-12:33:00
N1743 1 02/09/98-12:21:00
N1743 1 03/15/98-14:25:00
N1743 1 04/13/98-10:36:00
N2024 1 01/07/98-08:52:00
N2024 1 02/09/98-10:50:00
N2024 1 03/15/98-12:18:00
N2024 1 04/12/98-10:59:00
N2024 2 01/07/98-06:52:00
N2024 2 02/09/98-10:50:00
N2024 2 03/15/98-12:18:00
N2024 2 04/12/98-10:59:00
P1809 1 01/14/98-13:31:00
P1809 1 02/17/98-13:58:00
P1809 1 03/19/98-12:46:00
P1809 1 04/19/98-11:59:00
P1810 1 01/14/98-13:45:00
P1810 1 02/17/98-13:54:00
P1810 1 03/19/98-12:15:00
P1810 1 04/19/98-11:53:00
P1812 1 01/16/98-09:59:00
P1812 1 02/17/98-11:00:00
P1812 1 03/19/98-12:59:00
P1812 1 04/19/98-10:12:00
P1813 1 01/15/98-13:23:00
P1813 1 02/17/98-13:35:00
P1813 1 03/19/98-11:45:00
P1813 1 04/19/98-09:52:00
P1815 1 01/07/98-10:00:00
P1815 1 02/10/98-09:59:00
```

Output File

Using the Key Generator Programs (X810 or X820)

Now that you know how to use the Control Language, let's run through the steps for actually producing a file or report using the Key Generators. The following describes use of the Key Generators.

Summary — Using a Key Generator Program

1. Create the Key Generator Control File (TGX81A.CTL or TGX82A.CTL).
2. Create the optional Environment File.
3. Create the optional Output File.
4. Select the Record Definition File.
5. Run the Key Generator Program.

Note: The control file composer for the Key Generator will allow mixed-case to support XML. If you are not producing XML, you must enter only upper-case characters.

Step 1: Create the Key Generator Control File (TGX81A.CTL or TGX82A.CTL)

Use the Key Generator Control Language to specify your criteria for the output list or report. (See *Control Language — Logic and Structure* on page 15-3 if you need to review the use of this language).

```
KEY N1723,1,12/01/99-00:00:00
KEY N1725,1,12/01/99-00:00:00
KEY N1727,1,12/01/99-00:00:00
KEY N1729,1,12/01/99-00:00:00
KEY N1723,1,12/01/99-00:00:00
KEY N1725,1,12/01/99-00:00:00
KEY N1729,1,12/01/99-00:00:00
KEY N1723,1,12/01/99-00:00:00
KEY N1725,1,12/01/99-00:00:00
KEY N1729,1,12/01/99-00:00:00
```

Step 2: Add the (optional) Environment File

You can create the Environment File using the PRInt, CENtury, and TIMe commands:

PRInt [**ECOnomize** | **FULl**]

CENtury [**Yes** | **No**]

TIMe [**STAndard** | **ISO8601**]

TRAils

PRInt - (Optional) This command will instruct the program to write or not write format statement data into the output report (REPORT.HTML). Print Full is the default.

FULl - Instruct the program to write into the output report the format data records satisfying the search criteria in the Control File.

ECOnomize - Instruct the program *not* to write into the output report the format data records satisfying the search criteria in the Control File, resulting in a smaller report.

CENtury - If this command is specified, either with the Yes option or with none, all dates written by the Key Generator will contain a 4-digit year (e.g. 1999). If the command is absent or the No option is specified, the date will contain a 2-digit year (e.g. 99).

TIMe - This command controls how the format of date/time variables is printed. The default is STAndard (MM/DD/YY-hh:mm:ss). When ISO8601 is specified, all date/time variable outputs are in ISO8601-compatible format (MM/DD/YYYYThh:mm:ss).

TRAils - This command will direct the program to search information located within the Edit trails of edited cuts. Using this command and running the program in this mode may affect performance, as the program must retrieve twice the number of records from the database as compared to active cut records.

Step 3: Add the (optional) Keys File (Output File)

The Keys File is an input file that enables you to limit the records read and tested by the program. You may supply full keys, or you may omit the start time and supply only CUSTID and CHANNEL. (***Note*:** The Keys output file generated by an input program could be used for this Keys File.)

Step 4: Run the Key Generator Program

Submit the program for the database you are working with.

Key Generator Processing

The Key Generator Program starts testing each customer record with the first test statement in the control file. For each test statement, the value of the specified customer variable is compared to each test value in the value-list in succession. If the comparison between the customer variable and a test value matches the relation specified for that test value, the test statement is set to "TRUE" and no more test values are tried. If the value list is exhausted and no matching relationship has been found, the test statement is set to "FALSE".

After the test statement is set to "TRUE" or "FALSE", the action specified by the statement's true- or false-clause is taken. Test statements continue to be executed for the customer record until an execution true- or false-clause specifies "NEXT" or all the test statements have been performed.

The next cut is read, and the testing restarts from the top of the Control File, continuing in this manner until all customer records have been read or a "STOP" condition is reached. If a test-clause is not specified for a test statement, the statement is automatically set to "TRUE" and the true-clause is executed.

Chapter 16

Load Data Storage (X910 and X660)

Oracle Utilities Load Analysis includes a number of database maintenance procedures that may be performed as production processes by your System Administrator. This chapter includes a quick summary of these procedures in case you want to understand what they do.

This chapter also explains in detail a few of the programs Oracle Utilities provides for data management. With these programs, you can move cuts from the CLDB to the ALDB and back. In this chapter:

- **Scan, Archive/Delete Procedure (X910)**
- **Steps for Using the Scan, Archive/Delete Procedure to Transfer Cuts from CLDB to ALDB (X910)**
- **Steps for Using the Retrieval Program (X660)**

Periodically (probably once a month), you will need to move cuts from the CLDB to the ALDB. This makes the cuts available for analysis and keeps the CLDB to a manageable size. Oracle Utilities Load Analysis includes a special procedure for this task, as well as other procedures for other "housekeeping" — or database maintenance — activities.

Here is a summary of some of these procedures:

NAME	APPLICATION	FREQUENCY OF USE
Scan, Archive/ Delete (X910)	Identifies and transfers eligible cuts from CLDB to ALDB; deletes them from CLDB	Monthly
Retrieval (X660)	Copies cuts from the ALDB back to the CLDB, allowing you to reexamine and reedit data if necessary	As required

This chapter describes the procedures you will probably use most often — Scan, Archive/Delete and Retrieval. Your System Administrator will probably perform the other procedures.

> It is very important that you, your System Administrator, or your LAN support staff perform backups of all databases daily or weekly (frequency will depend on the volume of work). This helps minimize the amount of work lost in case of a hardware or software malfunction.

Scan, Archive/Delete Procedure (X910)

In this procedure, Oracle Utilities Load Analysis first scans the CLDB to determine which cuts are eligible for transfer to the ALDB. Only those cuts that are internally and externally valid, or have been flagged for merge and archive using the Load Data Editor Program, can be moved to the ALDB. This helps ensure that only complete, accurate data is made available for analysis.

Data transferred to the ALDB includes active records, inactive records, and edit trails. The transferred data is deleted from the CLDB. However, you can get it back using the Retrieval Program, described later in this chapter.

If a cut already exists in the ALDB with the same cut key as a transferred cut, the existing cut will be replaced.

Important Note about SET MERGE YES and SET ARCHIVE YES Edit Commands

Before transferring cuts to the ALDB, Oracle Utilities Load Analysis scans each cut in a series in chronological order, i.e., from earliest to most recent, to determine whether or not it is eligible for transfer. If a cut is internally and externally valid, Oracle Utilities Load Analysis will set its Merge flag to YES and will transfer it to the ALDB. Oracle Utilities Load Analysis rejects invalid cuts, cuts that follow invalid cuts, and cuts that precede or follow a missing cut. However, you can override these restrictions and make a cut eligible for transfer by forcing its Merge and Archive flags to YES using the SET Command in Load Data Editor (see *Chapter 9: Editing Data in the CLDB Using the Load Data Editor (X310 or X320)*). This is not usually recommended but may be necessary in some instances. For example, a cut that precedes a missing cut is automatically externally invalid, even if it is otherwise OK. In that instance, it is recommended that you force the cut's Merge flag.

There is an important distinction between the Merge and Archive flags that you should be aware of:

- SET ARCHIVE YES — If you force the Archive flag but not the Merge flag, the cut will be transferred to the ALDB, but it will not be available for analysis.
- SET MERGE YES — If you force the Merge flag, Oracle Utilities Load Analysis also automatically sets the Archive flag to YES. The cut will be moved to the ALDB and will be available for analysis.

Setting the Merge or Archive flags does not affect any of the other flags on a cut.

Oracle Utilities Load Analysis also retains a specified number of the most recent valid cuts in the CLDB (you define the number using a "retention parameter" in the Scan Environment File). This is done so that cuts will be available for external validation tests when new cuts are entered into the CLDB. *The Archive/Delete Program can handle up to 10,000 cuts in a cut series.*

More About How Cuts are Evaluated for Transfer

As described above, a cut is considered ineligible for transfer to the ALDB if it meets any one of the following conditions:

- Internally invalid
- Externally invalid (Keep in mind, a cut that precedes a missing cut is automatically externally invalid, even if it is otherwise OK)
- Follows an invalid cut
- Precedes or follows a gap.

Any of these restrictions can be overridden by forcing a cut's Merge or Archive flag to YES.

The following example illustrates the application of these rules.

CUT SERIES	VALIDATION STATUS INTERNAL	EXTERNAL	ARCHIVE FLAG
Cut 1	YES	YES	NO
Cut 2	YES	YES	NO
Cut 3	NO	YES	YES
Cut 4	YES	YES	NO
Cut 5	NO	NO	NO
Cut 6	YES	YES	NO
Cut 7	NO	NO	NO

In this example, all seven cuts are members of the same series and are listed in chronological order. Cuts 1, 2, and 4 are internally and externally valid and therefore qualified for transfer. Cut 3 is also eligible for transfer, even though it is internally invalid, because its Archive flag has been forced to YES. Cut 6 is ineligible, because Cut 5 has disqualified it. If the retention parameter in the Scan Environment File equals 1, the last valid cut in the series (in this case, Cut 4) will be retained. Therefore, just Cuts 1, 2, and 3 will be transferred.

Steps for Using the Scan, Archive/Delete Procedure to Transfer Cuts from CLDB to ALDB (X910)

Summary — Using the Scan, Archive/Delete Procedure

1. Create Scan Environment File (TGX91B.ENV).
2. Create Scan Control File (TGX91A.CTL) Archive/Delete Requests if you wish to transfer just selected cuts.
3. Create Archive/Delete Environment File (TGX92B.ENV).
4. Run Scan, Archive/Delete (X910).
5. **Check and KEEP output.**

   Note: It is CRITICAL that you keep the output.

Step 1: Create Scan Environment File (TGX91B) — required

Create the Scan Environment File using three commands:

RETain [*mm/dd/yy*[-*hh:mm:ss*] | *n* | 1]

SELect [**KEY** | **ALL**]

ARChive [**NORmal** | **FORCED**]

- **Retain** (Required) — Use the Retain Command to indicate the number or range of valid cuts that the program should leave in the CLDB for each cut series. It will leave the most recent cuts. You may specify either the number of valid cuts to be retained, or a retention date. If you choose the retention date option, any cuts that start *after* the date you specify will be retained; if you omit hours, minutes, and seconds, the end of the day (23:59:59) is assumed. For example, if you input "RET 2", the 2 most recent valid cuts for the specified series will remain in the CLDB and the rest will be moved to the ALDB. If you input "RET 03/18/99", any valid cuts with start times after 03/18/99-23:59:59 will remain in the CLDB. If you specify "RET" with no parameter, the default is "1".

- You should leave at least one cut per series in the CLDB, so that it is available for the External Validation tests.
- **Select** — Use the Select Command to indicate which cuts in the CLDB should be moved to the ALDB. "ALL", the default, indicates that you want all eligible cuts and cut series moved. Input the word "KEY" if you want to move only selected cuts or cut series. In that case, you *must* also supply a Control File consisting of a list of the cuts you want to move.
- **Archive** - Use the Archive command to select how the Scan program handles cuts that have been flagged for Archive (Archive Flag = Y). If you use "FORced" (the default), the program will force archive all cuts that are flagged for Archive regardless of any retention or disqualification criteria. If "NORmal" is used, cuts that are flagged for Archive will be treated as if they were normal valid cuts: subject to retention and other disqualification criteria.

  The following example illustrates the difference between "NORmal" & "FORced" archive modes with RETention parameter set to 2:

	Validation Status			Archive Mode	
Cut Series	**Internal**	**External**	**Archive Flag**	**FORced**	**NORmal**
CUT1	YES	YES	NO	Archived	Archived
CUT2	YES	YES	NO	Retained	Archived
CUT3	NO	YES	YES	*Force Archived*	Retained
CUT4	YES	YES	NO	Retained	Retained
CUT5	NO	NO	NO	Disqualified	Disqualified
CUT6	YES	YES	NO	Disqualified	Disqualified
CUT7	NO	NO	YES	*Force Archived*	Disqualified

```
RETAIN          1  /* RETAIN 1 VALID CUT
SELECT          ALL
ARChive         FOR
```

Figure 16-1 Example of a Scan Environment File

Step 2: Create Scan Control File (TGX91A) — optional

The Scan Control File contains generic keys of cut series to be scanned, and is *only required when "SELECT KEY" is specified* on the Scan Environment File. The purpose of the Scan Control File is to enable you to manage separate sets of cuts in the CLDB. For example, the CLDB could be segmented into survey data and power billing data with different retention parameters.

You can create the Control File using the Key Generator or by typing it. The format of each generic key is:

```
customer-id, channel
```

Blanks or commas may be used as delimiters.

```
A0001, 1       /* KEYS FOR SCAN
A0001, 2
B0012, 2
Z1234, 1
```

Figure 16-2 Example of a Scan Control File

Step 3: Create Archive/Delete Environment File (TGX92B)

Modify the Archive/Delete Environment File using three commands located in your Common\Data folder:

SOUrce {CLDB}

REPort [EXCeptions | ALL]

PRInt [ECOnomize | FULl]

- **Source** — Use the Source Command to indicate the origin of the cuts to be moved to the ALDB — CLDB must be specified.
- **Report** — This command enables you to define the mode of reporting. If you use "ALL" (the default), the program will print out the key and status of *every* cut moved to the ALDB and deleted from the CLDB. In the "EXC" mode, the program will print out *only* cuts that were missing or had key errors. This option can save significant time during program execution.
- **Print** — This command allows for the reduction of the amount of paper used for the Archive/Delete Execution Log. In the Economize mode, the dotted and blank separator lines between cuts are omitted. In the Full mode, the program prints the Archive/Delete Execution Log with full separation.

Step 4: Run Scan, Archive/Delete (X910)

Once you have created the necessary input files — Scan Environment File, Archive/Delete Environment File, and, optionally, the Scan Control File (if you used the KEY option) — you are ready to run the procedure. Use X910.

Scan Processing

The Scan Program examines each cut series in the CLDB designated for scanning, and determines which individual cuts are eligible for transfer. To be eligible, a cut must be:

- *internally and externally valid*, or
- *flagged for archiving or merging* by the Load Data Editor.

The Scan Program examines the cut series in chronological order; i.e., from earliest to most recent. *Valid cuts that occur after an ineligible cut are not candidates for archive unless flagged as such by the Load Data Editor.* After the Scan Program has identified the eligible cuts in a series, it *disqualifies the number of most recent ones as specified by the retention parameter.* The remaining cuts are thus identified for transfer.

If a cut has been flagged by the Editor using the "SET ARCHIVE YES" or "SET MERGE YES" commands, retention disqualification is not performed. (Remember that "SET MERGE YES" also sets the archive flag to "YES".)

If a gap exists in a cut series due to a missing cut, the cut preceding the gap will be externally invalid. The Scan Program therefore will not select that cut or the cuts after the gap for transfer unless their merge flags have been forced to "YES".

When the "KEY" mode is in effect, only those cut series whose keys appear in the Control File are scanned.

At the end of a successful run, the Scan Program produces a series of reports (described later in this chapter) and a Key File. The Key File lists all of the keys identified for transfer, and is automatically passed to the Archive/Delete Program.

Archive/Delete Processing

When "SOURCE CLDB" is employed, the Archive/Delete Program transfers each active CLDB record (along with its associated inactive record and edit trail, if any) to the ALDB. Each transferred cut is automatically deleted from the CLDB.

The Archive/Delete Program examines the status of the flags (merge, archive, internal validation and external validation) of each active cut identified in the Control File. To comply with the user's wishes the flags must be tested in a particular order. *If the merge flag has been set to "YES"* via the "SET MERGE YES" Command in the Load Data Editor, then the Archive/Delete Program will transfer the cut to the ALDB and delete it from the CLDB *without touching any of its other flags. If the archive flag has been set to "YES"* via the "SET ARCHIVE YES" Command in the Load Data Editor, then the Archive/Delete Program will transfer the cut to the ALDB and delete it from the CLDB *without touching any of its other flags. If neither the merge nor the archive flags are set,* the Archive/Delete Program will *test the validation flags. If both validation flags*, internal and external, are *valid*, then the program will *reset that cut's merge flag to "YES"* before it is written to the ALDB. The program will not *reset the merge flag to "YES"* under any other validation status situation.

This distinction regarding the setting of the merge flag is very important. Depending on whether or not the merge flag is set, the cut will or will not be extractable and in turn able to be analyzed in the Load Data Analysis Subsystem. *Only cuts with their merge flag set to "YES" can be extracted from the ALDB and analyzed.* So it is very important to keep this distinction in mind when archiving or extracting data. Also remember that *any cut whose archive flag has been set via the Load Data Editor will be archived, but cannot be merged and therefore will not be eligible for extraction or analysis*.

If an archived cut already exists in the ALDB (i.e., if a cut has been returned to the CLDB via the Archive Retrieval Program and its key has not been modified), *the transferred CLDB records will replace the existing ones in the ALDB.*

The Archive/Delete Program will *halt processing* if an edited cut's active record is transferred, but its *associated inactive record and/or edit trail cannot be found.*

Step 5: Check and *KEEP* Output

It is critically important that you keep the output reports. Remember, this procedure deletes cuts from the CLDB. These reports provide a record of what was deleted and when.

At the end of its run, the Scan, Archive/Delete procedure produces a REPORT.HTML file, with the following reports:

- Scan Environment Report
- Scan Execution Log (produced only if SELECT KEY is specified in the Scan Environment File)
- Scan Summary Log
- Archive/Delete Environment Report
- Archive/Delete Execution Log
- Archive/Delete Summary Log

Samples of these reports are reproduced on the following pages.

In addition, the program creates a file called "KEYS_ARCHIVED.CTL" which contains list of cut keys that have been successfully archived to the database.

Steps for Using the Retrieval Program (X660)

This procedure enables you to copy specified cuts in the ALDB and return them to the CLDB, so that you can reexamine and reedit them if necessary. The cuts in the ALDB are unaffected by the Retrieval Program. On the retrieved cuts, Oracle Utilities Load Analysis optionally sets the merge, archive, and external validation flags back to "NO".

If you request a cut that already exists in the CLDB, Oracle Utilities Load Analysis will reject the request.

Summary — Using the Retrieval Program
1. Create the Retrieval Control File (TGX66A.CTL) — a list of cuts to be retrieved.
2. Create the Retrieval Environment File (TGX66B.ENV).
3. Run Retrieval Program (X660).

Step 1: Create the Retrieval Control File (TGX66A.CTL)

The Control File is a list of the cuts you want to retrieve from the ALDB. You can enter just one request per line. You may use the Key Generator to produce this file or use a query list. Each request must be in the following format:

Customer-id, channel, [start-time]

You must specify the cuts in the same order that they appear on the ALDB Summary Log. (You can get this report using the procedure X460.)

Here is a sample Control File (Figure 16-1):

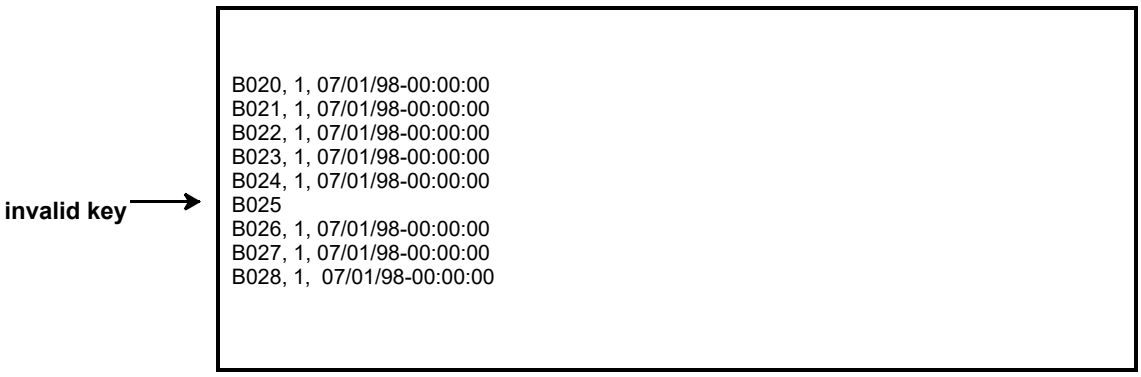


Figure 16-1 Sample Retrieval Control File

Step 2: Create the Retrieval Environment File (TGX66B)

Retrieval Environment File

There are four Environment File commands for the Retrieval program: SOURCE, FLAGS, SELECT, and DATE. The procedures delivered with Oracle Utilities Load Analysis refer to a pre-coded Environment File consisting only of the appropriate SOURCE Command.

Here is a sample Environment File :

DATe [*start-time stop-time* | **ALL** | **ALL** *stop-time* | *start-time*]

SOUrce [**ALDB**]

FLAgs [**NOReset** | **RESet**]

SELect [**ALL** | **KEY**]

Figure 16-2 Retrieval Environment File

Code the **SOU**rce Command as indicated; the source database for this Retrieval procedure is the ALDB.

You need to code the optional **FLA**gs Command, with **NOR**eset, only if you want the program to leave the merge and archive flags on the retrieved cuts unchanged from their status in the ALDB, rather than resetting them to "NO" (see above). The default is to reset the flags.

The **SEL**ect Command tells the program whether or not to process the Control File. SELect KEY, the default, instructs the program to retrieve only the Customer-ids listed in the Control File. The ALL option causes the program to loop through the entire database and select all the records that meet any of the other criteria set in the Environment File.

Step 3: Run Retrieval Program

Use X660.

Retrieval Processing

If a cut is found, the program will attempt to write the active, inactive, and edit trail records for that cut back to the CLDB. *If a cut being retrieved already exists on the CLDB, the retrieval request is rejected.*

Chapter 17

Using the Cut Series Gap Report Program for CLDB or ALDB (X490 or X491)

The Cut Series Gap Report lists, for each cut series (customer id and channel), the start date and time of the series, any gaps in the data, and the stop date and time of the series. The report shows the start and stop times of each gap. A gap is defined both as a time period between the end of a cut and the beginning of the next cut in the series, and as a period of missing data (as defined by status codes) within a single cut. The Gap Report may be produced for all cut series in the CLDB or ALDB, or for a specified portion of the database. It reports on active cuts only.

Steps for Using the Cut Series Gap Report Program (X490 or X491)

The following pages describe the steps for using the Cut Series GAP Report Program using either the CLDB or ALDB Summary Reporter Procedures.

Summary — Using the Cut Series Gap Report Program

1. Determine whether you want to test the entire database for gaps or selected cut series. If you want a report on the entire contents of the database, go to Step 3. If you want to report only selected cut series, create the Control File (TGX49A.CTL) — a list of cut series keys.
2. Create the Environment File (TGX49B.ENV) — optional reporting period and other parameters.
3. Run the Cut Series Gap Report Program (X490 for CLDB, X491 for ALDB).
4. Check output.

Step 1: Create the Control File (TGX49A.CTL) to select specific cut series

If you want to report the entire contents of the database, go to Step 3. The Control File (Figure 17-1) is a list of the cuts series you want to report. You can create the file manually or with the CLDB or ALDB Key Generator Program.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Each report request has the following format (this is a cut series request):

Customer-id, channel-number

Enter one request per line. Oracle Utilities Load Analysis will sort the Control File prior to running the program if submitted by the client GUI. You can separate the two elements of a request (customer-id and channel-number) with blank spaces and/or a comma.

If you do not supply a Control File, the program will consider all cuts in the database eligible for testing and reporting.

Important note:

If you do supply a Control File, it will only be used if you specify SELECT KEY in the Environment File.

```
B001,1
B002,1
B002,2
B003,1
INVALIDCHANNEL,32768
B004,1
B005,1
```

Figure 17-1 Sample Control File

Step 2: Create the Environment File (TGX49B.ENV)

Use the Environment File (Figure 17-2) to specify an optional date range for the reporting period, and other parameters.

Create the Environment File with these six commands:

DATe [*start-time stop time* | **ALL**]

SELect [**KEY** | **ALL**]

SOUrce {**CLDB** | **ALDB**}

QUAlity ['*q*' | '8']

PRInt [**GAP** | **ALL**]

TITle *optional-user-title*

- **Date** — Use this optional command to set the start- and stop-time of the reporting period for the *selected cuts*. You define "selected cuts" using the Select Command described below — either all cuts in the database or just those listed in the Control File.
- There are several ways to specify the reporting period:
- If you specify just ALL, then all selected cuts will be tested for gaps, regardless of their start-time. (This is the default.)
- If you supply both a start-time and a stop-time, all selected cuts that contain intervals at or between the two times will be tested.
- If you supply just a start-time, all selected cuts that contain intervals at or after the specified start-time will be tested.
- If you specify ALL followed by a stop-time, the program will test all selected cuts that contain intervals at or before the specified stop-time.

```
DATE  ALL
SEL   KEY
SOU   CLDB
```

Figure 17-2 Sample Environment File for the CLDB Gap Reporter

To specify a date/time, use the "mm/dd/yy-hh:mm:ss" format. If you omit the "hh:mm:ss" portion, the start- and stop-times will default to 00:00:00 and 23:59:59, respectively.

- **Select** — Use this optional command to select the cut series you want tested for gaps. "ALL" indicates that you want every cut in the database that falls within the specified date range to be tested. "KEY" indicates that you want only those customers whose keys appear in the Control File and fall within the desired date range to be tested. ALL is the default.
- **Source** — Use this command to identify which database you want the cuts reported from. **This is a required command** and you **must** input "CLDB" if you are using the CLDB Cut Series Gap Report Program (X490), "ALDB" if you are using the ALDB Cut Series Gap Report Program (X491).
- **Quality** — Use this optional command to define what intervals within a cut will ***not*** be considered gap intervals. Input the worst acceptable status code. Oracle Utilities Load Analysis will treat any interval with a status code worse than 'q' as missing, i.e., as constituting part of a gap.

  Note: The status code must be input with apostrophes (') on either side. The default is '8' (only status '9' missing intervals constitute a gap or gaps within a cut).

- **Print** — Use this optional command to determine which of the tested cut series are to be reported. If you specify "PRInt GAP", only cut series found to contain one or more gaps will be listed in the Cut Series Gap Report; all series without any gaps will be omitted from the report. If you specify "PRI ALL" or "PRI" (no option), or if you omit this command, all tested cut series will be reported.
- **Title** — This optional command allows you to specify one additional title line on the Cut Series Gap Report and Gap Report Summary Log. The title begins with the first nonblank character following the TITLE keyword and may be up to 196 characters long; it is not enclosed in quotes.

The example for the ALDB Cut Series Gap Report Program is identical, except "SOURCE ALDB" is used.

Step 3: Run the Cut Series Gap Report Program

Once you have created the desired inputs, you are ready to submit the job. Use the X490 Submit screen for the CLDB or the X491 Submit screen for the ALDB.

Cut Series Gap Report Processing

When "SELECT ALL" mode is used, each cut on the database that falls within the desired date range is checked for gaps, both within it and between it and adjacent cuts in the same series.

When "SELECT KEY" mode is used, only those cuts whose keys appear in the Control File and within the desired date range are checked. If a key is not found, an appropriate error message is generated and processing continues with the next Control File record.

Chapter 18

Using the Cut Series Overlap Report Program for CLDB or ALDB (X530 or X531)

The Cut Series Overlap Report lists, for each cut in a database, the start time, stop time, and total number of recorded intervals in the cut, plus the start time of any cut in the same series (customer ID and channel) with intervals overlapping the given cut. The report shows the start and stop times of each overlap period, and the number of overlapping intervals. Two cuts are defined as overlapping if intervals recorded in the first cut extend beyond the first interval in the second cut. The Overlap Report may be produced for all cut series in the CLDB or ALDB, or for a specified portion of the database. It reports on active cuts only.

Overlaps in data may produce undesirable results; as cuts are merged together, Oracle Utilities Load Analysis's merging rules are to use data that is from the most recent start time and assign the worst status code to the resultant overlapped interval.

Steps for Using the Cut Series Overlap Report Program (X530 or X531)

The following pages describe the steps for using the Cut Series Overlap Report Program using either the CLDB or ALDB Overlap Reporter procedures.

Summary — Using the Cut Series Overlap Report Program

1. Determine whether you want to test the entire database for overlaps or selected cut series. If you want a report on the entire contents of the database, go to Step 3. If you want to report only selected cut series, create the Control File (TGX53A.CTL) — a list of cut series keys.
2. Create the Environment File (TGX53B.ENV) — optional reporting period and other parameters.
3. Run the Cut Series Overlap Report Program.

Step 1: Create the Control File (TGX53A.CTL) to Select Specific Cut Series

If you want to report the entire contents of the database, go to Step 3. The Control File (Figure 18-1) is a list of the cut series you want to report. You can create the file manually, generate it with the Customer Data Extraction Program, or create it with the CLDB or ALDB Key Generator Program.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Each report request has the following format (this is a cut series request):

```
Customer-id, channel
```

Enter one request per line. Oracle Utilities Load Analysis will sort the Control File before running the program if submitted by the client GUI. You can separate the two elements of a request (customer-id and channel) with blank spaces or a comma.

If you do not supply a Control File, the program will consider all cuts in the database eligible for testing and reporting.

Important note:

If you do supply a Control File, it will only be used if you specify SELECT KEY in the Environment File.

```
B0001,1
B0002,1
B0003,1
B003A,2
B0004,1
C0002,1
A0001,2
```

Figure 18-1 Sample Control File

Step 2: Create the Environment File (TGX53B.ENV)

Use the Environment File (Figure 18-2) to specify an optional date range for the reporting period, and other parameters.

Create the Environment File with these five commands:

DATe [*start-time stop time* | **ALL**]

SELect [**KEY** | **ALL**]

SOUrce [**CLDB** | **ALDB**]

PRInt [**OVErlaps** | **ALL**]

TITle *optional-user-title*

TOLerance *number of overlapping intervals to tolerate*

OUTput [CSV | FIX]

- **Date** — Use this optional command to set the start- and stop-time of the reporting period for the *selected cuts*. You define "selected cuts" using the Select Command described below — either all cuts in the database or just those listed in the Control File.

  There are several ways to specify the reporting period:

  - If you specify just ALL, then all selected cuts will be tested for overlaps, regardless of their start-times. (This is the default.)
  - If you supply both a start-time and a stop-time, all selected cuts that contain intervals at or between the two times will be tested.
  - If you supply just a start-time, all selected cuts that contain intervals at or after the specified start-time will be tested.
  - If you specify ALL followed by a stop-time, the program will test all selected cuts that contain intervals at or before the specified stop-time.

```
DATE  ALL
SEL   KEY
SOU   CLDB
```

Figure 18-2 Sample Environment File for the CLDB Overlap Reporter

  To specify a date/time, use the "mm/dd/yy-hh:mm:ss" format. If you omit the "hh:mm:ss" portion, the start- and stop-times will default to 00:00:00 and 23:59:59, respectively.

- **Select** — Use this optional command to select the cut series you want tested for overlaps. "ALL" indicates you want every cut in the database that falls within the specified date range to be tested. "KEY" indicates that you want only those customers whose keys appear in the Control File and fall within the desired date range to be tested. ALL is the default.
- **Source** — Use this command to identify which database you want the cuts reported from. **This is a required command** and you **must** input "CLDB" if you are using the CLDB Cut Series Overlap Report Program (X530), "ALDB" if you are using the ALDB Cut Series Overlap Report Program (X531).
- **Print** — Use this optional command to determine which of the tested cuts are to be reported. If you specify "PRInt OVE", only cuts found to be overlapped by other cuts will be listed in the Cut Series Overlap Report; all cuts that are not overlapped will be omitted from the report. If you specify "PRI ALL" or "PRI" (no option), or if you omit this command, all tested cuts will be reported.

- **Title** — This optional command allows you to specify one additional title line on the Cut Series Overlap Report and Overlap Report Summary Log. The title begins with the first nonblank character following the TITLE keyword and may be up to 196 characters long; it is not enclosed in quotes.
- **Tolerance** - This optional command allows you to specify the number of overlapping intervals to tolerate. Overlaps will only be reported if the number of overlapping intervals exceeds the tolerance amount. The default tolerance level is 1.

  The example for the ALDB Cut Series Overlap Report Program is identical, except "SOURCE ALDB" is used.
- **Output** - This optional command allows you to create an interface file, and specify the format of the file. The file produced will be named Gaps.csv or Overlap.csv if CSV is selected, or Gap.dat or Overlap.dat if FIX is selected.

  When used with the Gap reporter, the CSV option produces a CSV file in the following format:

  <ID>,<Channel>,<gap Start time>,<gap Stop time>

  The FIX format will produce:

Field	Dimension/Type	Offset
Recorder ID	Char(64)	1 - 64
Channel	Char(5)	65 – 69
Gap Start	Char(17) MM/DD/YY HH:MM:SS	70 – 86
Gap End	Char(17) MM/DD/YY HH:MM:SS	87 - 103

  When used with the Overlap reporter, the CSV option will produce a file following this format:

  <ID>,<CH>,<OL'dCutStart>,<OL'dCutStop>,<OL'ingCutStart>,<OL'ingCutStop>,<OLStart>,<OLStop>,<OL'ing#Ints>

  The FIX option will produce a file following this format:

Field	Type/Dimension / Format	Offset / Position
Recorder ID	Char (64) Alphanumeric	1 – 64
Channel	Integer (5) Pos Numeric	65 - 69
OL'd Cut Start	Char (17) MM/DD/YY HH:MM:SS	70 – 86
OL'd Cut Stop	Char (17) MM/DD/YY HH:MM:SS	87 - 103
OLing Cut Start	Char (17) MM/DD/YY HH:MM:SS	104 – 120
OLing Cut Stop	Char (17) MM/DD/YY HH:MM:SS	121 - 137
OL Start	Char (17) MM/DD/YY HH:MM:SS	138 – 154
OL Stop	Char (17) MM/DD/YY HH:MM:SS	155 – 171
OL Num Ints	Integer (14) Pos Numeric	172 - 188

Step 3: Run the Cut Series Overlap Report Program

Once you have created the desired inputs, you are ready to submit the job. Use the X530 Submit screen for the CLDB or the X531 Submit screen for the ALDB.

Cut Series Overlap Report Processing

When "SELECT ALL" mode is used, each cut on the database that falls within the desired date range is checked to determine whether it is overlapped by a prior cut in the series.

When "SELECT KEY" mode is used, only those cuts whose keys appear in the Control File and within the desired date range are checked. If a key is not found, an appropriate error message is generated and processing continues with the next Control File record.

Gap and Overlap Reporter Output Interface File

An interface file, triggered via an environment command, can be produced from the Overlap and Gap reporter programs. This command option will control the enablement of this behavior as well as the format of the file.

Environment file: (for both Gap and OLR)
OUT CSV | FIX

Chapter 19

Late Cut Reporter

The Late Cut Reporter assists electric utilities in identifying cuts that have not been loaded into the Oracle Utilities Load Analysis database. Occasionally, a translation of data will not be entered into Oracle Utilities Load Analysis for reasons such as:

- **A cut was not included in the load data file for Oracle Utilities Load Analysis**
- **The load data file was not entered into Oracle Utilities Load Analysis**
- **Problems occurred during the translation process.**

In many instances, the missing data can be recovered if the translation department is notified in time.

The Late Cut Reporter reports late cuts by identifying channels whose data ends before a specified cutoff. For example, on a monthly meter reading cycle, all translations should be entered into Oracle Utilities Load Analysis within 35 days. Given a cutoff date 35 days prior to the present date, the Late Cut Reporter will examine each cut series to determine when the data ends. For each series that ends before the cutoff, all cuts are listed with their start and stop times. This information allows the utility to identify the recorder and approximate start time of the missing cut corresponding to the last month.

The Late Cut Reporter may be run for the entire database or a specified list of customers. Therefore, key lists and cutoff dates may be created to correspond to billing cycles.

This program may be set up to run on a scheduled basis using the Oracle Utilities Load Analysis Sequencer.

Topics covered in this chapter are:

- **Program Usage**
- **Late Cut Reporter Inputs**
- **Late Cut Reporter Processing**
- **Late Cut Reporter Outputs**
- **Operating Procedures**
- **Special Considerations**

Program Usage

The Late Cut Reporter reports late cuts by identifying channels whose data ends before a specified cutoff time. This enables the utility to monitor the progress of load data translation and ensure that data is not lost or overlooked. The Late Cut Reporter can be run as needed on the entire database or for selected customers. It may also be scheduled using the Sequencer. The latter capability allows the user to synchronize the cutoff time with the customer billing cycle. Following is a complete description of program usage for the Late Cut Reporter.

Late Cut Reporter Inputs

The Late Cut Reporter requires three inputs:

- **The Current Load Database (CLDB)**
- **An Environment File**
- **A Control File.**

The Environment File designates the mode of execution (i.e., select all cut series or specific cut series before a particular cutoff time). The Control File contains the user specified keys to locate specific cut series. This section describes in detail the contents of the Environment and Control files.

The *Environment File* contains two commands that determine the mode of execution:

DATe *cut-off time*

SELect [**ALL** | **KEY**]

> **Note:** If the program is being run through the sequencer, you may elect to create a script that runs directly before the Reporter to create the Environment File with a specific DATe command.

The DATe Command determines the endpoint, or cutoff-time, of a cut series; any cut series within the SELECT range whose stop-time falls on or before the given cutoff time will be reported. If the cutoff time is given in the form "mm/dd/yy", then it will be interpreted as time "mm/dd/yy-00:00:00"; that is, the beginning of that day. Note that the Late Cut Reporter will not report cuts missing in the middle of a cut series (a condition flagged during cut series validation). Failure to specify a valid DATe Command will result in a fatal execution error.

The SELECT Command determines the set of customer keys to process. If "SELECT ALL" is specified, the entire CLDB is searched for a late cut series. Only those series specified in the Control File are searched when "SELECT KEY" is chosen. The default mode is "SELECT KEY".

An example of an Environment File is:

DATE 12/05/88

SELECT KEY

The *Control File* contains a list of cut series to be examined for meter stop-times. There is one request per line; commas or blanks can separate the parameters. Each line should correspond to the format:

Customer-id, *channel*

The Control File is sorted before processing.

Note: This program supports embedded SQL commands. See **Querying the Database with Embedded SQL** on page 4-7 for more information. This program supports pre-process key generator. See **Using the Preprocess Key Generator in a Control File** on page 4-4 for more information.

Late Cut Reporter Processing

Late Cut Reporter processing begins by checking the syntax of the Environment File. Syntax errors detected in the DATE Command will result in an appropriate error message being written to the Environment Report followed by abnormal termination of the program. Syntax errors detected in the SELECT Command will activate the default "SELECT KEY" mode of execution.

Following successful validation of the Environment File commands, program execution will proceed in one of two execution modes. The "SELECT ALL" mode will cause the entire CLDB to be read sequentially. All cut series whose last cut stop time is on or before the user specified date and time will be written to the Late Cut Reporter Execution Log. Alternately, the "SELECT KEY" mode will choose only those cuts specified in the user supplied Control File. If an empty Control File is used with the "SELECT KEY" mode of execution, it will result in an appropriate message being written to the Execution Log and the program will abnormally terminate. During normal execution, keys not found in the CLDB (invalid keys) will be flagged and reported at the end of the Execution Log.

After all user requests have been processed, a summary report will be produced containing the execution statistics.

Late Cut Reporter Outputs

The Late Cut Reporter Program produces four output reports:

- Environment Report
- Execution Log
- Summary Report.
- An interface file of Late Keys (LATEKEYS.DAT)

The Environment Report, Execution Log, and Summary Report are described below, and are contained in the REPORT.HTML file in your job directory:

Environment Report — The Late Cut Environment Report indicates the mode of execution, selected cutoff time, and prints any error messages encountered during syntax checking.

Execution Log —The Late Cut Execution Log lists the cut series ending on or before the user specified cutoff time. In the "SELECT KEY" mode of execution, invalid keys will also be printed at the end of this report if they exist.

Summary Report —The Late Cut Summary Report will list execution statistics concerning both valid and invalid input/output operations.

Operating Procedures

The Late Cut Reporter searches the CLDB for cut series that stop before a specified cutoff time. The cutoff time represents the billing cycle time when translations should be entered into the Oracle Utilities Load Analysis database. For each cut series that falls short, the last stop-time is reported to help the user identify the start of the next cut that is missing.

Special Considerations

The cut series keys (i.e., customer-id, channel) are sorted before processing by the Late Cut Reporter Program. This will result in two files in your job directory: one with your original cut series keys, and one sorted.

No Existing Cuts in Database

The Late Cut Reporter produces a comma separated values file consisting of control file requests, which are cut series requests for which no cuts exist in the database. This file has two columns and will be named KeysNotFound.csv, and contains a list of Recorder id's and channels or Id's and categories that were not found in the database.

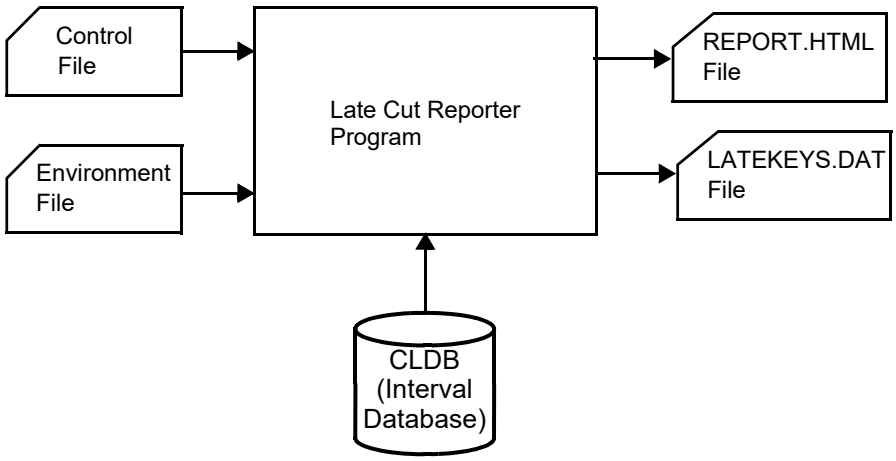


Figure 19-1 Late Cut Reporter Program

Appendix A

Oracle Utilities Unit of Measure Codes

Intervals may be summed or averaged when aggregating depending on the unit of measure. The aggregation technique is noted on the table.

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
01	kWh	Sum
02	kW	Average
03	kVARh	Sum
04	kVAH	Sum
05	Temperature (ºF)	Average
06	KQD	Average
07	V^2H (PTP)	Sum
08	kQh	Sum
09	kQh (45 degrees)	Sum
10	I^2H	Sum
11	Volts	Average
12	Amps	Average
13	Temperature (ºC)	Average
14	Dewpoint	Average
15	Amplitude	Average
16	Miscellaneous	Sum
17	Minute Run Time (MRT)	Sum
18	Wind Velocity (Cm/Sec)	Average
19	Fraction V2H (PTN)	Average
20	Percent	Average

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
21	Flow	Average
22	kVAR	Average
23	kVA	Average
24	kVA Ratio	Average
25	Power Factor	Average
26	Hertz	Average
27	Feet	Average
28	Minutes	Sum
29	On/Off (Tap position)	Average
30	Inches	Average
31	Individual kWh	Sum
32	kWh r	Sum
33	Individual Totalized kVARh	Sum
34	kVARh r	Sum
35	Individual Totalized Temperature (°F)	Average
36	kVAh r	Sum
37	IND V^2H (Individual totalized V^2H)	Sum
38	IND kQh (Individual totalized kQh)	Sum
39	KQH r	Sum
40	Miscellaneous	Average
41	IND Volts (Individual totalized Volts)	Average
42	IND Amps (Individual totalized Amperes)	Average
43	IND Temperature (°C) (Individual totalized temperature, degrees Celsius)	Average
44	Mw	Sum
45	MVAR	Average
46	MVA	Average
47	IND MRT (Individual totalized MRT)	Sum
48	IND CMS (Individual totalized CMS)	Average
49	Run Hours	Sum
50	Equivalent Full Load Hours	Sum
51	KWH-OUT	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
52	KW-OUT	Average
53	KVARH-OUT	Sum
54	KVAH-OUT	Sum
55	KQH-OUT	Sum
56	LEAD-KVARH	Sum
57	LEAD-KVARH-OUT	Sum
58	LAG-KVARH	Sum
59	LAG-KVARH-OUT	Sum
60	Gallons / Minute	Average
61	BTU	Sum
62	THERMS	Sum
63	Cubic Feet / Minute	Average
64	Cubic Feet / Second	Average
65	WM^2	Average
66	Relative Humidity	Average
67	MPH	Average
68	THI	Average
69	Gallons	Sum
70	Cubic Feet	Sum
71	Temperature Difference	Average
72	KVAR-OUT	Average
73	KVA-OUT	Average
74	Knots	Average
75	Degrees	Average
76	CCF (Hundred cubic feet)	Sum
77	CF / Hour	Average
78	Pounds / Square inch	Average
79	Dollars	Sum
80	DECATHERMS	Sum
81	Pounds	Sum
82	Pounds / Hour	Average
83	MPOUNDS	Sum

CODE	DESCRIPTION	AGGREGATION TECHNIQUE
84	MPOUNDS/Hour	Average
85	$/KWH	Average
86	$/MW	Average
87	$/MWH	Average
88	$/Hour	Average
89	Volt Hours	Sum
90	Individual Totalized Cubic Feet	Sum
91	Individual Totalized Btu	Sum
92	Pressure in MILLIBARS	Average
93	Visibility in Miles	Average
94	Cents per kWh	Average
99	Individual Totalized Gallons	Sum
100	MWH	Sum
102	Euros	Sum
103	Euros per MWH	Average
104	Euros MW	Average
105	GW	Average
106	TWH	Sum
107	Cubic Meters (M3)	Sum
108	Mega Joules per Cubic Meter (MJ/m3)	Average
109	Kilograms per Cubic Meter (Kg/m3)	Sum
110	Cubic Meters per Hour (M3/H)	Average

UOM Compatibility

When merging cuts with different units-of-measure, cuts with specific differing UOMs can be combined while others can't. For example, you can't merge two cuts if the UOM of one cut is inches and the UOM of the other is dollars, because whichever UOM is assigned to the merged cut wouldn't apply to at least some of the data.

However, cuts with certain specific different UOMs can be combined. UOMs that can be combined are referred to as compatible UOMs, and are listed in the table below.

Compatible UOMs
01,31,32,51
02,52
03,33,34,53,56,57,58,59

Compatible UOMs
04,36,54
05,35
07,37
08,09,38,39,55
11,41
12,42
13,43
17,47
18,48
22,72
23,73
61,69
69,99
70,90

These units of measures may have their intervals divided by the IPH for display:

Demand-Type UOMs
02, 05, 06, 07, 10, 22, 23, 24, 52, 72, 73, 105, 110

Appendix B

Record Formats Used by Oracle Utilities Load Analysis Input Programs

The Oracle Utilities Load Analysis Direct Input programs require input records to be in one of the Oracle Utilities record formats.

- **Standard Format** — This is the most commonly used format for the Oracle Utilities Load Analysis Direct Input Program. This format is created by the Oracle Utilities Load Analysis Direct Output Program as well as several Data Translation systems developed by other vendors.
- **Enhanced Format** — This is a high-precision format developed for Oracle Utilities systems released after 01/01/2000. See **Appendix A**: **Oracle Utilities Enhanced Input/Output Interval Data Format** in the *Oracle Utilities Load Analysis Configuration Guide* for details on the .LSE format.

Standard Format

Input blocks consist of the following types of records:

- First header record, which includes sort code, customer-id, channel, start-time, stop-time, intervals-per-hour, unit-of-measure code, and an alternate format flag.
- Second header record, which includes a sort code, meter-start-reading, meter-stop-reading, meter and pulse offsets, meter multiplier, and, if the alternate format flag is set to '0', the pulse multiplier.
- Third header record, which includes a sort code, the first half of the customer descriptor, if the alternate format flag is set to '1', the pulse multiplier, and if a statistic record, the population and weight.
- Fourth header record, which includes a sort code and the second half of the customer descriptor.
- Data records, which include a sort code, load data values, and corresponding status codes for each interval of data in the cut.

All input records have a fixed record length of 80 bytes. To form a single CLDB cut, a set of records consisting of one each of the first, second, third, and fourth header records, and a variable number of data records, in that order, is required. Each data record holds 12 intervals of load data. There can be as many data records as necessary to create the CLDB cut, provided the number of data records does not exceed 875 records for any one cut. In other words, *a cut can hold approximately 30 days of 5-minute data, 90 days of 15-minute data, or one year of 60- minute-per-interval data.*

Detailed descriptions of the five types of records and their formats are given below.

First Header Record

The First Header Record must contain a sort code of 0001 as the first four bytes. The Customer Identifier is a 20-character field. The Channel is a single digit from 0 to 9. The start-time and stop-time are in the form "mmddyyhhmm". This is the only date-time format supported in the INP format. Interval start times are "minute beginning," meaning that 12:00:00 is represented by 12:01 in this format. End times are interval ending. For example, 23:59:59 (end of the day) is represented as 24:00. The beginning of the day is represented by mmddyy0001. Intervals-per-hour must be either 1, 2, 4, 12, or 60.

The Unit-of-Measure Code is a 2-character field that identifies the unit-of-measure of the data. *Appendix A: Oracle Utilities Unit of Measure Codes* lists standard Unit-of-Measure codes.

The Alternate Format Flag is a 1-byte field that indicates the location and the format of the pulse multiplier. If the flag is '1', the value of the pulse multiplier is less than 1.000 and is located in the Third Header Record. If the flag is '0', the pulse multiplier's value is greater than 0.9999 and is located in the Second Header Record.

Second Header Record

The Second Header Record must contain a Sort Code of 0002 as the first four bytes. The Meter-Start-Reading and Meter-Stop-Reading are 7-digit fields with one implied decimal. The Meter Multiplier and Pulse Multiplier are 15-digit fields with five implied decimals. The Meter Offset and Pulse Offset are assigned 15-digit fields (a total of 16 bytes) with five implied decimal digits. A Second Header Record must immediately follow a First Header Record.

Third Header Record

The Third Header Record must contain a Sort Code of 0003 as the first four bytes. The Descriptor is a 40-character field used as the first half of the 80-character descriptor of the associated CLDB cut. The Alternate Pulse Multiplier is a 15-digit field with 15 implied decimal digits. The Population is a 9-digit field with no implied decimal. The Weight is a 12-digit field with five implied decimals. A Third Header Record must immediately follow a Second Header Record.

Fourth Header Record

The Fourth Header Record must contain a Sort Code of 0004 as the first four bytes. The Descriptor is a 40-character field used as the second half of the 80-character descriptor of the associated CLDB cut. A Fourth Header Record must immediately follow a Third Header Record.

Data Record

A Data Record must contain a Sort Code between 1000 and 9999, inclusive, as its first four bytes. A variable number of Data Records normally appears following the four header records. The Sort Code of the first Data Record in that set should be 1000, and the Sort Code of each subsequent Data Record must be incremented by 1. Every interval of every Data Record will be entered into a CLDB cut. If the last Data Record does not have 12 recorded intervals, then the last Data Record must have trailing pad intervals, indicated by data value 0 and status code "9".

Each Data Record has an array of 12-element pairs, with each element pair containing a data value and a status code. Data values are 5-digit numbers and must be padded with leading zeros. See Figure 3-2 for a description of status codes.

A set of Data Records is terminated by either a First Header Record, a break in the sequence of Sort Codes, or end of file.

Table B-1 through Table B-5 describe the five types of records.

In the following tables, 'PIC' attributes represent numbers, for which '9' represents numeric positions, numbers within parentheses are repetition factors, and 'V' indicates the position of the decimal point. For example:

```
PIC'999999V9' and PIC'(6)9V9'
```

are equivalent, and represent a number with 6 places to the left of the decimal and 1 place to the right.

'CHAR' attributes represent character fields that contain both alphabetic and numeric data. The length of the field is specified by the number in parentheses.

Table B-1: First Header Record Format

ELEMENT	DESCRIPTION	ATTRIBUTES	LENGTH IN BYTES
1	Sort Code	PIC '9999'	4
2	Customer Identifier	CHAR(20)	20
3	Channel	PIC '9'	1
4	Start Time	PIC '(10)9'	10
5	Stop Time	PIC '(10)9'	10
6	Intervals-per-hour	PIC '99'	2
7	Unit-of-Measure	PIC '99'	2
8	Alternate Format	PIC '9'	1
9	Filler	CHAR(30)	30

Table B-2: Second Header Record Format

ELEMENT	DESCRIPTION	ATTRIBUTES	LENGTH IN BYTES
1	Sort Code	PIC '9999'	4
2	Meter Start Reading	PIC'(6)9V9'	7
3	Meter Stop Reading	PIC '(6)9V9'	7
4	Meter Multiplier	PIC '(10)9V(5)9'	15

5	Pulse Multiplier	PIC '(10)9V(5)9'	15
6	Meter Offset	PIC 'S(10)9V(5)9'	16
7	Pulse Offset	PIC 'S(10)9V(5)9'	16

Table B-2: Second Header Record Format

ELEMENT	DESCRIPTION	ATTRIBUTES	LENGTH IN BYTES
1	Sort Code	PIC '9999'	4
2	Descriptor	CHAR(40)	40
3	Alternate Pulse Multiplier	PIC 'V(15)9'	15
4	Population	PIC '(9)9'	9
5	Weight	PIC '(7)9V(5)9'	12

Table B-3: Third Header Record Format

ELEMENT	DESCRIPTION	ATTRIBUTES	LENGTH IN BYTES
1	Sort Code	PIC '9999'	4
2	Descriptor	CHAR(40)	40
3	Filler	CHAR(36)	36

Table B-4: Fourth Header Record Format

ELEMENT	DESCRIPTION	ATTRIBUTES	LENGTH IN BYTES
1	Sort Code	PIC '9999'	4
2	Interval(12) Load Data Array Status Array	PIC '99999' CHAR(1)	12 * (6) 5 1
3	Filler	CHAR(4)	4

Table B-5: Data Record Format

Appendix C

Key Generator Variable Lists

This appendix documents all of the variables you can use to create Control Files for the Key Generator programs.

VARIABLE NAME	ATTRIBUTES	DESCRIPTION	RESTRICTIONS ON VALUES
ARCHIVE	FLAG	Determines whether or not cuts can be archived using X910	May be either "YES" for OK to archive, or "NO" for not OK to archive
CHANNEL	INTEGER(2)	Channel-number	May be 0 through 32767
COUNTm	INTEGER	Used to keep a running count of the occurrence of specified conditions as the program applies your test statements to the records.	See **Counter Variables** on page 15-11 for more information.
COUNTINT	INTEGER	A counter variable used with the INTERVAL variable to return the count of intervals that pass the INTERVAL test function.	See **Counter Variables** on page 15-11 for more information.
COUNTINT++	INTEGER	Supplement to the COUNTINT variable that increases the COUNTINT counter by adding to it the number of intervals that pass a subsequent INTERVAL test function.	See **Counter Variables** on page 15-11 for more information.
COUNTSTAT	INTEGER	A counter variable used with the STATUS variable to return the count of intervals that pass the STATUS test function.	See **Counter Variables** on page 15-11 for more information.
COUNTSTAT++	INTEGER	Supplement to the COUNTSTAT variable that increases the COUNTSTAT counter by adding to it the number of intervals that pass a subsequent STATUS test function.	See **Counter Variables** on page 15-11 for more information.
CUSTID	CHAR(64)	Customer-id	
DESC	CHAR(80)	Descriptor	Up to 80 characters
DST	CHAR(1)	DST Participant Flag	May be either "Y" or "N"

VARIABLE NAME	ATTRIBUTES	DESCRIPTION	RESTRICTIONS ON VALUES
EDITBYRS	CHAR(1)	Indicates whether cut has been edited by Oracle Utilities Rules Language	May be either "Y" for edited or "N" for not edited
EDITED	CHAR(1)	Indicates whether cut has been edited	May be either "Y" for edited or "N" for not edited
ESUM	REAL(8)	Energy Sum	Total of energy for all intervals in cut
EXTVALID	CHAR(1)	Indicates whether cut has passed external validation tests	May be either "Y" for passed or "N" for not passed
INTERVAL	REAL(8)	Returns TRUE for each cut that contains interval values that match user-specified criteria.	See **Comparison Test Statements for Intervals and Statuses** on page 15-7 for more information.
INTVALID	CHAR(1)	Indicates whether cut has passed internal validation tests	May be either "Y" for passed or "N" for not passed
MERGE	CHAR(1)	Determines whether cuts can eventually be extracted from the ALDB to the ELDB for analysis	May be either "Y" for OK to extract or "N" for not OK to extract
MMULT	REAL(8)	Meter Multiplier	See note for PMULT
MOFFSET	REAL(8)	Meter Offset	See note for PMULT
MSGi	CHAR(80)	Used to test for validation messages, 1-10, i is the number of the message	Up to 80 characters
MSG*	CHAR(1)	Used to concatenate all messages together for the test	
MSTART	REAL(8)	Meter-start-reading	See note for PMULT
MSTOP	REAL(8)	Meter-stop-reading	See note for PMULT
ORIGIN	CHAR(1)	Origin value of the cut	May be one of "M", "P", "S" (ELDB only), or "C"
PMULT	REAL(8)	Pulse Multiplier (Documentation field only)	To obtain full precision of fields defined as "REAL", it's necessary to use a field-spec; e.g., pulse multiplier should be specified as PMULT 6.2
POFFSET	REAL(8)	Pulse Offset (Documentation field only)	See note for PMULT
POP	REAL(8)	The statistic population assignment	XZO
RECTYPE	CHAR(1)	Record Type	May be either ' ' for active (either unedited or most recent version of edited cut, or 'A' for inactive (old original version of an edited cut) or 'T' for edit trail
SPI	INTEGER	Seconds Per Interval	86400, 3600, 1800, 900, 300, 60
START	DATE/TIME	Cut start-date and -time	Format: MM/DD/YY-HH:MM:SS
STARTDAT	DATE	Cut start date	Format: MM/DD/YY

VARIABLE NAME	ATTRIBUTES	DESCRIPTION	RESTRICTIONS ON VALUES
STARTHR	INTEGER(4)	The hour value of the start time	
STARTMIN	INTEGER(4)	The minute value of the start time	
STARTSEC	INTEGER(4)	The second value of the start time	
STATUS	REAL(8)	Returns TRUE for each cut that contains status codes that match user-specified criteria.	See **Comparison Test Statements for Intervals and Statuses** on page 15-7 for more information.
STOP	DATE/TIME	Cut stop-date and -time	Format: MM/DD/YY-HH:MM:SS
STOPDAT	DATE	Cut stop date	Format: MM/DD/YY
STOPHR	INTEGER(4)	The hour of the stop time	
STOPMIN	INTEGER(4)	The minute of the stop time	
STOPSEC	INTEGER(4)	The second of the stop time	
TRAILS	CHAR(1)	Defines all 40 available TRAILs in a single string	
TRAILCT	CHAR(1)	Defines the number of edit trails associated with a cut	
TRAIL1, TRAIL2.. TRAILn… TRAIL40	CHAR(1)	Defines specific edit trails up to the 40th edit trail	
TOTINT	INTEGER(4)	Total number of intervals	
TZSN	CHAR(32)	Time Zone Standard Name	Should be a value defined in the LSCALENDAR.CFG.XML file
UOM	INTEGER(2)	Unit of Measure code	Must be one of allowable codes (see *Appendix A: Oracle Utilities Unit of Measure Codes*)
VALFLAGE	CHAR(1)	Indicates cut failed X210 Energy test	May be either " " (passed) or "E" (failed)
VALFLAGI	CHAR(1)	Indicates cut failed X210 test	May be either " " (passed) or "I" (failed)
VALFLAGO	CHAR(1)	Indicates cut failed X210 Outage test	May be either " " (passed) or "O" (failed)
VALFLAGN	CHAR(1)	Indicates failed X210 test	May be either " " (passed) or "N" (failed)
WEIGHT	REAL(8)	Statistic weight assignment	$0 \leq X \leq 1$

Appendix D

Glossary

Term	Definition
Active Record	A cut in the CLDB used for validation, editing, reporting, plotting, and merging
Aggregated Interval	An interval formed by adding two or more successive intervals; if insufficient data is available to complete the aggregated interval, it is assigned a status code "7" (partial)
ALDB	Archive Load Database
Archive Flag	A field on a CLDB record indicating that the cut is to be archived to the ALDB
AXDB	Auxiliary Database, used to support Load Data Input Programs by providing selection, translation and descriptor modification
Battery Carryover	A recorder feature that allows recorders to continue functioning during power outages
Billing Period	The cycle of time between meter readings for billing
Channel	One of several end-use recordings from the same recorder/ microprocessor
CLDB	Current Load Database
Completion Report	A summary report issued by a translator after a cartridge has been processed; similar information appears on the Load Data Reporter
Computed Intervals	The number of intervals between two dates, calculated by the formula: ((stop-time - start-time)/seconds-per-interval) + 1
Computed Stop Time	Time derived by adding intervals to a known date, at a specified intervals per hour
Customer-id	Unique alphanumeric identifier for each customer
Cut	A record in the CLDB or ALDB containing descriptive and load data for one customer/channel/start-time combination
Cut Series	All the records in the CLDB for the same customer/channel
Demand Data	Energy data multiplied by the recorded intervals-per-hour

Term	Definition
Descriptor	A field on a CLDB record usually containing name and address information
Edit Block	A sequence of editing operations that apply to a single cut
Edit Trail	Record maintained in the CLDB for each edited cut containing all the edit commands applied to the original load data
End-Use Class	The ultimate usage associated with a cut of load data (e.g., electric hot-water heating, space heating, lighting, etc.)
Energy Data	An interval data value
Energy Sum	Sum of all interval values in a cut
Expected Duration	Time length of cut based on the difference between its start and stop-times
Expected Intervals	Computed intervals between two dates, adjusted for daylight savings time
External Validation Tests	Sequence of tests that determine whether or not a cut matches up with the following cut in its series
External Validation Flag	Field on a CLDB record indicating its external validation status
False Interval	Data interval based on possibly erroneous timing information
Inactive Record	Backup record of the original, unedited data of an edited cut
Insertion	Editing process in which additional data intervals are placed between existing data intervals
Internal Validation Tests	Sequence of tests on a single cut's data to determine its internal consistency
Internal Validation Flag	Field on a CLDB record indicating its internal validation status
Interval	A quantity of energy data recorded for a discrete period of time
Intervals-Per-Hour	The frequency of individual intervals of data
Key	The identification fields for a database record
Key Command	An edit command identifying a cut for editing
Loss of Potential	A status code indicating the presence of a power outage
Merge Flag	A field on a CLDB or ALDB record indicating that it is available to be merged to the ELDB for analysis
Meter Energy	Net-meter-reading times the meter multiplier plus the meter offset times the number of data intervals
Meter Multiplier	Multiplicative constant used to convert a meter reading to engineering units
Meter Offset	Additive constant used to convert a meter reading to engineering units
Meter Rollover	Condition that occurs when the maximum value on a meter dial is reached and the dial starts over again at zero

Term	Definition
Meter-Start-Reading	Meter dial reading at the beginning of a recording period
Meter-Stop-Reading	Meter dial reading at the end of a recording period
Minutes-Per-Interval	Duration of individual intervals of data
Missing Interval	Data interval containing erroneous timing information
Net-Meter-Reading	The meter-stop-reading minus the meter-start-reading adjusted for meter rollover
Non-Normal Code	Status code indicating a data interval of questionable quality
Overwrite	An editing process in which data intervals are replaced
Patching	An editing process in which load data for insertion or overlay is obtained from a cut in the CLDB
Peak	Maximum load data value over a known time period
Proration	Editing process in which load data is scaled proportionally to match the meter energy and adjusted for meter rollover
Pulse Data	Raw data corresponding to revolutions of a meter disk or some other pulse initiator
Pulse Energy	Pulse data times the pulse multiplier plus the pulse offset
Pulse Multiplier	Multiplicative constant used to convert pulse data to engineering units
Pulse Offset	Additive constant used to convert pulse data to engineering units
Recorded Duration	Time length of cut based on the number of recorded intervals
Recorded Intervals	Number of data intervals actually present for a cut
Recovery Mode	Translator mechanism for processing tape cartridges without using timing intervals
Seconds-Per-Interval	Duration of individual intervals of data
Short Interval	Incomplete data interval usually associated with first or last data interval
SIC	Standard Industrial Classification code
Splice Peak	Data interval (obtained by adding the last short interval of one cut with the first short interval of the next cut in the series) whose value exceeds both preceding and following intervals; replaced by the average of the preceding and following intervals
Start-Time	The date and time of the beginning of cut
Status Code	Special field associated with each data interval indicating source or quality of load data
Stop-Time	The date and time of the end of a cut
Translator	Device for converting cartridge data to computer-readable format
Unit-of-Measure	A code indicating engineering units of the data
(UOM) Code	

Term	Definition
Validation Message	Diagnostic message issued by the Validation Program
Validation Message Area	An area in each cut's CLDB record where validation messages associated with the cut are stored

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

X

Z