CIMS Lab, Inc.

CIMS Server for UNIX®

Installation and User Guide

Version 2.1

CIMS Lab Publication Number: CS-UNIX-210-02

Published 02/06/06

Mailing Address

CIMS Lab, Inc.
3013 Douglas Blvd., Suite 120
Roseville, CA 95661-3842

Table of Contents

Preface

Keeping track of IT usage can be a formidable task for any organization. Most sizable enterprises consist of many platforms, systems, and subsystems, each costly to own, maintain and operate. In addition, each component has its own unique record format and metrics.

With CIMS Server for UNIX®, you can process, access, and analyze IT resource usage metrics from many IT resources and platforms in one centralized location. CIMS Server for UNIX helps you to better understand who is using IT resources and how the resources are being used.

CIMS Server for UNIX allows you to allocate, distribute, or charge IT costs to users, cost centers, and organizations in a manner that is fair, understandable, reproducible, and easy to administer.

The technology behind CIMS Server for UNIX is based on CIMS Lab's many years of experience in the development and implementation of Resource Accounting, Capacity Planning, and IT Chargeback products.

About CIMS Lab

Founded in 1974, CIMS Lab has focused on meeting the financial and resource reporting requirements of Information Services Departments. CIMS has evolved with corporate IT management requirements. Focused commitment to client service and support sets CIMS apart from competing products. Our goal is to provide the best chargeback and resource reporting software in the world at the lowest possible cost to our customers.

CIMS Lab strongly believes in and executes the concept of continuous product improvement. Customers have access to CIMS product development personnel to ensure that customer feedback and other critical issues are incorporated into the next release of the product.

Contacting CIMS Lab

To contact CIMS Lab with questions, comments or problems, please use one of the following methods:

For product assistance or information:

USA & Canada, toll free - (800) 283-4267
International - (916) 783-8525
FAX - (916) 783-2090
World Wide Web - http://www.cimslab.com

Mailing Address:

CIMS Lab, Inc.
3013 Douglas Blvd., Suite 120
Roseville, CA 95661-3842

About This Guide

This guide describes the installation procedures and administration functions and features of CIMS Server for UNIX.

The following table describes the chapters in this guide. If you are installing and setting up CIMS Server for the first time, you should begin with *Chapter 1, Installing CIMS Server for UNIX and Getting Started* before continuing to the other chapters in the guide.

Ch. No.	Chapter Name	Content Description
1	*Installing CIMS Server for UNIX and Getting Started*	Provides an overview of the CIMS Server for UNIX system and steps for installation and getting started.
2	*System Architecture*	Describes the architecture of CIMS Server for UNIX.
3	*Processing Data*	Provides an overview of the data processing cycle and components and describes how to run CIMS Server for UNIX.
4	*Database Administration*	Provides information about setting up and using the database.
A	*Control Statements*	Describes the control statements used by CIMS Processing Engine.
B	*Running the CIMS Server for UNIX Install Script*	Provides an example log of a typical CIMS Server for UNIX installation.

Conventions

Some or all of the following conventions appear in this guide:

Symbol or Type Style	Represents	Example
Alternate color	hyperlinked cross-references to other sections or chapters in this guide; if you are viewing this guide online, you can click the cross-reference to jump directly to its location	...see Chapter 3.
Italic	words that are emphasized	...the entry *after* the current entry...
	a new term	...called a *source object.*
	the titles of other manuals	*CIMS Server Administrator's Guide*

	variables in file names or system names	`Job`*`definitionname`*`.pl`
		`yyyymmdd``.txt`

Bold	names of interface items such as tabs, boxes, buttons, lists, and check boxes.	Select the **CPU Value** check box Type the database name in the **Database Name** box. Click **Edit**.
`Monospace`	directories, file names, command names, computer code, computer screen text, system responses, command line commands, what the user types	`processes` directory `nightly.sh` script type `$CIMS_HOME/bin/CIMSAdmin`
< >	the name of a key on the keyboard	Press `<Enter>`
▶	navigating a menu or a folder	Customer Area ▶ Product Downloads

Related Publications

This guide is intended to be used in conjunction with the *CIMS Server Administrator's Guide*. In addition, you might find it helpful to have these additional guides available for reference:

- *CIMS Server Web Reporting User's Guide*
- *CIMS Data Collector for UNIX User Guide*
- *CIMS Mainframe Data Collector and Chargeback System User Guide*
- *CIMS Server for DB2 Database Setup Guide*

1

Installing CIMS Server for UNIX and Getting Started

This chapter provides installation and configuration instructions for CIMS Server for UNIX.

About CIMS Server for UNIX

CIMS Server for UNIX is a component of the multi-platform CIMS Server product suite. CIMS Server for UNIX supports UNIX, Linux® zLinux, and UNIX System Services (USS) on z/OS and includes the following key features:

- Supports an Oracle database on a UNIX/Linux or Windows® platform. (Also supports SQL Server on Windows and DB2 on z/OS.)
- Provides the CIMS Processing Engine programs (CIMSAcct, CIMSBill, and CIMSLoad) as Java™ programs that can be run natively on UNIX/Linux. This enables you to use the UNIX/Linux platform to consolidate and process usage data from multiple IT resources and platforms into a common output format for costing and reporting. This output is database ready and may be used for multiple purposes such as reporting and as a feed to internal or third-party systems.

  CIMS Server for UNIX processes usage data associated with:

  - Operating systems
  - Databases
  - Network systems
  - Storage systems
  - Internet systems
  - E-mail systems
  - Other applications or monitors that create usage metering data

As shown in Figure 1-1 on page 1-3, it is useful to think of CIMS Server for UNIX as a funnel that accepts usage data and returns organized information that can help IT managers and staff to track and allocate resources.

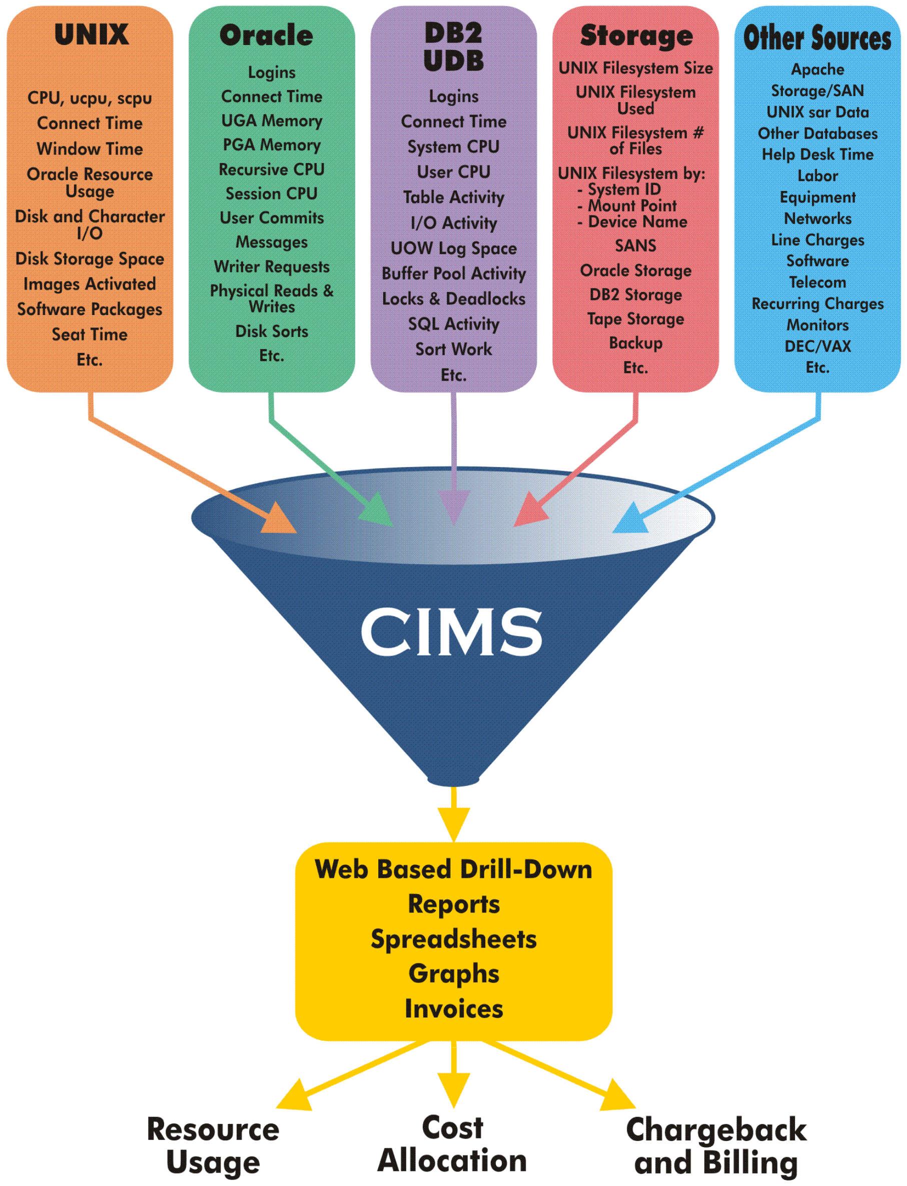


Figure 1-1 • CIMS Collects Usage Data and Organizes It As Reporting Information

CIMS Server for UNIX Component Overview

The following components support the CIMS Server for UNIX system:

- CIMS Data Collectors.
- The CIMS Server for UNIX application server.
- A database server.

> **Note • CIMS Server for UNIX and the database can reside on the same server or on separate servers. In general, a configuration involving separate servers might offer users better performance.**

- The CIMS Server Administrator application.
- The CIMS Server Web Reporting application.

The following sections describe each of these components.

CIMS Data Collectors

CIMS Data Collectors read and convert usage metering data generated by applications (usually standard usage metering files such as log files) and produce a common output file that is used by CIMS Server for UNIX.

There are two types of collectors that are used with CIMS Server for UNIX: external collectors and internal.

- External collectors are CIMS Data Collectors for Windows, Mainframe, and UNIX. These collectors collect usage metering data and produce CIMS Server Resource (CSR) files or CIMS Server Resource Plus (CSR+) files, which can be sent to CIMS Server for UNIX for processing. External collectors are described in detail in the CIMS documentation for each collector (see *Related Publications* on page ix).
- Internal collectors are included with CIMS Server for UNIX. These collectors collect and convert the data from usage metering files to produce CSR files. These CSR files can be processed by CIMS Server for UNIX, CIMS Server, or CIMS Mainframe.

  For more information about architecture of the internal collectors, see *Collection Files (collectors Directory)* on page 2-2.

About CSR and CSR+ Files

CSR files are created by CIMS Data Collectors for Windows, CIMS Data Collector for UNIX, and CIMS Mainframe Data Collector and Chargeback System (11.6 and earlier). CSR files can also be created by many other third-party applications.

CSR+ files are created by CIMS Mainframe Data Collector and Chargeback System 12.0 and later.

The format of the CSR and CSR+ file records is the same, with the exception that the records in the CSR+ file contain an additional header at the beginning of the record.

For a complete description of the CSR and CSR+ file formats, refer to the *CIMS Server Administrator's Guide*.

CIMS Server for UNIX Application Server

The CIMS Server for UNIX application server contains the following key components:

- CIMS Processing Engine
- CIMS Processing Engine Administrator
- CIMS Processing Scripts

CIMS Processing Engine

CIMS Processing Engine is a fast, efficient data processing engine capable of processing and applying business rules to large volumes of data. CIMS Processing Engine consists of the following Java programs:

- **CIMSAcct.** This program processes the CSR or CSR+ files provided by the external or internal collectors and performs functions such as account code conversion and shift code determination. CIMSAcct produces a CSR+ output file, which is sorted by the CIMSSort subroutine and sent to CIMSBill.
- **CIMSBill.** This program processes the sorted CSR+ file from CIMSAcct and builds output files that contain the billing information that is used to generate invoices and reports.
- **CIMSLoad.** This program loads the output files from CIMSBill into an Oracle, SQL Server, or DB2 for z/OS database.

For more information about the CIMS Processing Engine components, see *Chapter 3, Processing Data*.

Important! • CIMS Processing Engine is available as a Java-based engine for the UNIX platform or a COM-based engine for the Windows platform (see *Using CIMS Processing Engine* on page 4-2 for more information). In this guide, CIMS Processing Engine refers to the engine for the UNIX platform unless noted otherwise.

CIMS Processing Engine Administrator

CIMS Processing Engine Administrator is a Java GUI application used to maintain the CIMS Server for UNIX configuration files. For more information about this application, see *Set Up the Configuration Files* on page 1-13.

CIMS Processing Scripts

CIMS processing scripts are Perl-based scripts that automate the task of processing of data from various feeds and loading output data into the database.

Database Server

CIMS Server for UNIX uses any of the following databases:

- Oracle on UNIX or Windows
- SQL Server on Windows
- DB2 on z/OS

The tables in the database contain the administrative and configuration data needed for chargeback and reporting as well as the output records that are produced by CIMS Processing Engine. The database tables are described in detail in the *CIMS Server Administrator's Guide*.

For more information about setting up and using the database, see *Chapter 4, Database Administration*.

CIMS Server Administrator

CIMS Server Administrator is a Windows-based GUI application that is used to maintain the information in the CIMS Server for UNIX database. The following are some of the key tasks that are performed using CIMS Server Administrator:

- Setting up and maintaining rates, clients, and CIMS Server Web Reporting users and groups.
- Specifying the reports that are used for CIMS Server Web Reporting and setting up reporting options such as batch reporting.
- Viewing and maintaining the data in the database, including archiving, purging, and restoring usage output records that have been loaded to the database.
- Setting configuration and administration options for the CIMS Server Web Reporting Web site.

For more information about using CIMS Server Administrator, refer to the *CIMS Server Administrator's Guide*.

CIMS Server Web Reporting

The CIMS Server Web Reporting application provides comprehensive cost accounting, chargeback, and resource reporting in an easy-to-use, browser-based environment. CIMS Server Web Reporting includes the following features that ensure that users receive the data they need in a clear, user-friendly format:

- **Drill down.** CIMS Server Web Reporting invoices and many other reports include drill down that enables users to view detailed cost and usage information.
- **Multiple report formats.** CIMS Server Web Reporting provides reports in report, graph, and spreadsheet format.

- **User customization options.** CIMS Server Web Reporting users can set up their own list of favorite reports, spreadsheets, and graphs. Users can also create reports "on-the-fly" within the CIMS Server Web Reporting Web site and publish reports for review by other users.

For more information about using CIMS Server Web Reporting application, refer to the *CIMS Server Web Reporting User's Guide*.

System Specifications

The following are the suggested server specifications for the CIMS Server for UNIX system.

- For the CIMS Server for UNIX application server:
  - 100 MB available hard drive space for the base installation.
  - 5 GB available hard drive space for the process definition data. Required space depends on the number of process definitions, the size of the daily job logs, and the length of time that data is retained in the system.

    Note • Process definitions, job logs, and other components of the CIMS Server for UNIX system are described in *Chapter 2, System Architecture*.

  - 1 GB of memory.
- For the database server, refer to the Oracle, Microsoft®, or IBM® documenation for the database.

Pre-Installation Setup

Before installing CIMS Server for UNIX, make sure that you have installed and configured the following third-party software and drivers:

- Java2 Standard Edition Runtime Environment (JRE) 1.4 or later. The JRE is available free of charge from Sun Microsystems.
- Perl 5
- JDBC drivers for Oracle, SQL Server, or DB2 for z/OS database access. For Oracle and SQL Server, JDBC drivers are available free of charge from Oracle and Microsoft. For information about obtaining the DB2 JDBC driver, refer to the IBM documentation for the Type 4 Universal JDBC driver.

Installing the Java 2 Standard Edition Runtime Environment

The J2SE Runtime Environment (JRE) is required to run CIMS Processing Engine. If your system does not have the J2SE JRE installed or the installed J2SE JRE is a release earlier than 1.4, you can download the JRE from Sun Microsystems at http://java.sun.com/j2se/1.4.2/download.html.

You will be prompted to enter the path to the J2SE JRE during the CIMS Server for UNIX installation.

Installing Perl

Perl 5 ships with most common UNIX systems (e.g., AIX, HP-UX, Solaris, Linux). If Perl 5 is not installed on your computer, you can get the Perl source and binary distributions at www.cpan.org/ports.

By default, CIMS Server for UNIX accesses Perl at `/usr/bin/perl`. If the Perl executable is not in this location, you need to create a soft link to the executable in `/usr/bin`. For example:

```
-s /usr/local/bin/perl /usr/bin/perl
```

If the Perl executable is not in `/usr/bin/perl`, you will be asked if you want to create a soft link to the executable during installation.

Installing JDBC Drivers

JDBC is an application program interface (API) specification for connecting programs written in Java to the data in a wide range of databases. To enable CIMS Processing Engine to access an Oracle, SQL Server, or DB2 z/OS database, the appropriate JDBC drivers must be available.

You can download the Oracle and Microsoft JDBC drivers from the Oracle and Microsoft Web sites free of charge. For information about obtaining the DB2 JDBC driver, refer to the IBM documentation for the Type 4 Universal JDBC driver.

CIMS Server for UNIX assumes that you will be using the Oracle, Microsoft, or DB2 drivers. However, you can use other drivers, such as DataDirect. In this case, you must specify that you want to override the default driver as described on page 1-15.

Installing the Oracle JDBC Driver

The JAR file that contains the native Oracle JDBC driver is `ojdbc14.jar`. If you are installing CIMS Server for UNIX on a computer that has an Oracle database installed, this driver is already installed. If you are installing on a computer that does not have an Oracle database, you can copy the `ojdbc14.jar` file from another computer or you can download the file from http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html.

Select the `ojdbc14.jar` file without the tracing information.

Setting up Oracle JDBC Driver Access

CIMS Processing Engine requires access to the `ojdbc14.jar` file. You need to place the file in the CIMS `$CIMS_HOME/jlib` directory or place a link to the file in the `jlib` directory.

Installing the SQL Server JDBC Drivers

The JAR files that contain the native SQL Server JDBC driver support are `mssqlserver.jar`, `msbase.jar`, and `msutil.jar`. To download these drivers, go to http://www.microsoft.com/downloads/details.aspx?FamilyID=07287b11-0502-461a-b138-2aa54bfdc03a&DisplayLang=en to get SQL Server 2000 for JDBC Service Pack 3.

If the preceding link is no longer valid, go to http://www.microsoft.com/downloads and search for the JDBC driver. Click **SQL Server 2000 Driver for JDBC SP3**.

Setting up SQL Server JDBC Driver Access

CIMS Processing Engine requires access to the `mssqlserver.jar`, `msbase.jar`, and `msutil.jar` files. You need to place these files in the CIMS `$CIMS_HOME/jlib` directory or place a link to the files in the `$CIMS_HOME/jlib` directory.

Installing the DB2 for z/OS JDBC Driver

The JAR file that contains the native DB2 JDBC driver is `db2jcc.jar`. For the steps required to set up this file, refer to the *CIMS Server for DB2 Database Set Up Guide*.

Installing Other JDBC Drivers

To install another JDBC driver, refer to the documentation for the driver. CIMS Processing Engine requires access to the driver JAR file. You need to place the file in the CIMS `$CIMS_HOME/jlib` directory or place a link to the file in the `jlib` directory.

Preparing to Install CIMS Server for UNIX

Before you install CIMS Server for UNIX, follow the procedures in the following table. Depending on your processor's speed, current system load, etc., installation time is approximately 10 to 15 minutes.

Task	Description
Perform Backups	**If you are installing CIMS Server for UNIX:** Although CIMS Server for UNIX does not interfere with or modify the UNIX system, it is a good precautionary measure to ensure that the system has been recently backed up before you install or upgrade any system level product. **If you are upgrading CIMS Server for UNIX:** Before performing an upgrade, back up the CIMS Server for UNIX database and directories so you can recover to the original state, if necessary. In addition, back up any CIMS scripts that you have modified.
Verify That You Have Enough Disk Space for Installation	Make sure you have enough disk space for installing CIMS Server for UNIX. The application and files require 20 to 25 MB. The additional space requirement is dependent on the size of the accounting files generated on the computer.

Table 1-1 • Pre-installation Procedures

Task	Description
Obtain the CIMS Server for UNIX License Key	Make sure you have a CIMS Server for UNIX license key. You must enter this information after installation. If you do not have a license key, contact CIMS Lab (see *Contacting CIMS Lab* on page viii).
Create a CIMS User Account	CIMS Server for UNIX is a system level product that operates in conjunction with the UNIX operating system. You have the option of having root maintain CIMS Server for UNIX or setting up a CIMS user account. The advantage of a user account is that the CIMS Server for UNIX administrator does not need root privileges on a daily basis for maintenance. CIMS Lab recommends that you use the user account name `cims`, but you can use any name.
Create a CIMS Group (Optional)	Creating a CIMS Server for UNIX GID is recommended. CIMS Lab suggests that you use the group name `cims`, but you can use any name.
Create a Script Log for Installation	Script the installation for analysis in case a problem occurs during the install. To create a script log, use the following command: `> script  install.log` Type `exit` or press `<ctrl-c>` to exit the script session after the installation is complete. For an example script log, see *Appendix B, Running the CIMS Server for UNIX Install Script*.

Table 1-1 • Pre-installation Procedures (Continued)

Installing or Upgrading CIMS Server for UNIX

Note • Before you begin the CIMS Server for UNIX installation, make sure that you have performed the steps in the *Pre-Installation Setup* section beginning on page 1-7.

To install or upgrade CIMS Server for UNIX, follow the steps in this section.

Log On

CIMS Server for UNIX installation requires root privilege to create directories and execute privileged commands. Log on to the UNIX system using the Super-User (root) account.

Get the CIMS Server for UNIX Distribution Files

You need the following distribution files to install CIMS Server for UNIX:

- `CS_cims_install` (the install script)
- One of the following tar files (contains CIMS executables, scripts, sample processes, etc.)
  - For ASCII UNIX platforms, `CS_dist.tar`
  - For z/OS USS platform, `CS_zOS_dist.tar`

You can get the distribution files from the CIMS Product CD or the CIMS Lab Web or FTP site as described in the following sections.

From the CIMS Product CD

To get the CIMS Server for UNIX distribution files from the CIMS Product CD:

1. Load and mount the CD using the appropriate commands for your system.
2. Copy the files from the `UNIX/CS_Unix` folder on the CD to a temporary directory.

From the CIMS Lab Web Site

To get the CIMS Server for UNIX distribution files from the CIMS Lab Web site, go to the Customer Area ▸ Product Downloads page and download the files to a temporary directory.

You need your CIMS Server for UNIX license key to access this page. If you do not have a license key, contact CIMS Lab (see *Contacting CIMS Lab* on page viii).

From the FTP Site

To get the CIMS Server for UNIX distribution files from the CIMS FTP site:

1. Contact CIMS Lab to obtain your FTP access user name and password (see *Contacting CIMS Lab* on page viii).
2. FTP to `ftp.cimslab.com`.
3. Change the directory to `ntunix/CS_unix`:

   ```
   ftp> cd ntunix/CS_unix
   ```

4. Get the CIMS Server for UNIX distribution files and place them in a temporary directory:

   ```
   ftp> bin
   ftp> get CS_cims_install
   ftp> get CS_dist.tar
   ```

   Note: The tar file for z/OS USS is `CS_zOS_dist.tar`.

5. Disconnect from the FTP site:

   ```
   ftp> bye
   ```

Run the CS_cims_install Script

The `CS_cims_install` script is in your temporary working directory. This script must be executed by root. The script will prompt you for the following:

- The install location for CIMS Server for UNIX. The default is `/usr/cims`.
- The path to the directory where the CIMS Server for UNIX distribution files are located.
- The name of the CIMS account.
- The name of the CIMS account group.
- The path to the J2SE JRE `bin` directory.

After you install CIMS Server for UNIX, you need to set up the `cims.par` and `CIMSDBConfig.xml` configuration files as described in *Set Up the Configuration Files* on page 1-13. These files are located in the `$CIMS_HOME/config` directory and provide the parameters needed to run CIMS Server for UNIX.

Getting Started

This section takes you through the steps required to set up and use CIMS Server for UNIX. You should follow these steps in the order presented.

To help you to get started quickly, this section does not contain the detailed information found in other chapters in this guide. Where applicable, references to more detailed information are provided.

Create a Database

CIMS Server for UNIX uses an Oracle database on UNIX or Windows; a SQL Server database on Windows; or a DB2 database on z/OS.

To create an Oracle database, consult your Oracle DBA. For more information about the Oracle database requirements, see *Chapter 4, Database Administration*.

To create a SQL Server database, follow the steps in *CIMS Server Administrator's Guide*.

Important! • Although the *CIMS Server Administrator's Guide* provides instructions for creating a Windows authenticated user login for SQL Server, you cannot use Windows authentication if you are using the UNIX version of CIMS Processing Engine. You must use a SQL Server authenticated user login.

To create a DB2 database, consult your DB2 DBA. For more information about the DB2 database requirements, see the *CIMS Server for DB2 Database Set Up Guide*.

Set Up the Configuration Files

Before the CIMS Server for UNIX Processing Engine can process data, you must set up the configuration files `CIMSDBConfig.xml` and `cims.par`. The `CIMSDBConfig.xml` and `cims.par` files provide parameters required to connect to the database and run CIMS Processing Engine, respectively.

The `CIMSDBConfig.xml` and `cims.par` files are built in the `$CIMS_HOME/config` directory during installation, but the files must be updated before they can be used. To update these files you need to use the CIMS Processing Engine Administrator GUI program (`CIMSAdmin.jar`) *or* the CIMSDBConfig utility (`CIMSDBConfig.jar`).

CIMS Processing Engine Administrator provides an easy-to-use interface for modifying the `CIMSDBConfig.xml` and `cims.par` files; however, the GUI requires an XTERM server. If you are running CIMS Server for UNIX on z/OS USS and do not have an XTERM server, you can use the CIMSDBConfig utility to modify these files.

Important! • Do not rename the `CIMSDBConfig.xml` or `cims.par` file or move the files from the `config` directory.

Modifying the CIMSDBConfig.xml File

The CIMS Processing Engine components (CIMSAcct, CIMSBill, and CIMSLoad) access the database through a JDBC interface. The `CIMSDBConfig.xml` file contains the parameters needed to connect to one or more databases within your organization. Each database must have a CIMS Data Source entry in the file that points to the database.

You can have multiple CIMS Data Sources, but you must specify one as the default data source. The default data source is stored in the `cims.par` configuration file.

The following sections provide steps for modifying the `CIMSDBConfig.xml` file using the CIMS Processing Administrator GUI or the CIMSDBConfig utility (see page 1-18).

Using CIMS Processing Administrator to Modify the CIMSDBConfig.xml File

To start CIMS Processing Engine Administrator:

At the command prompt, type:

```
$CIMS_HOME/bin/CIMSAdmin
```

Because CIMS Processing Engine Administrator is a Java GUI application, you need to define the `DISPLAY` environment. If `DISPLAY` is not defined in your environment, you can supply it on the command line as shown in the following example:

```
> $CIMS_HOME/BIN/CIMSAdmin -d roxie:0
```

The CIMS Processing Engine Administrator main window appears. To access the CIMS Data Source definitions, click **Data Source Configuration** to open the CIMS Server JDBC Data Source Configuration dialog box. Follow the steps in the following sections to add, edit, or remove a data source, and to set a data source as the default.

To add a CIMS Data Source:

1. Click **Add** to add a CIMS Data Source entry to the `CIMSDBConfig.xml` file.

   The CIMS Server JDBC Data Source Definition dialog box appears.

2. Edit the CIMS Data Source settings as follows:

   - **Data Source ID.** Type an ID for the CIMS Data Source (maximum of 32 characters). You can use the database name as the data source ID or use another ID.

   - **Database Name.** The name of the database that you are creating the CIMS Data Source for (maximum of 32 characters).

   - **Database Schema.** If you are using an Oracle or DB2 database, the name of the database schema. If you are using a SQL Server database, the database owner prefix (usually `dbo`).

   - **Database Type.** Click the database type (Oracle, SQL Server, or DB2).

- **Database Port.** The TCP/IP server port that is assigned to the database server. The default port is dependent on the database:
  - For Oracle, the default port is 1521
  - For SQL Server, the default port is 1433
  - For DB2, the default port is 446
- **Database Server.** The domain name or IP address of the database server.
- **User ID and Password.** The user ID and password for the database.

Note • If you are using SQL Server, you must use a SQL Server-authenticated user ID. You cannot use a Windows-authenticated user ID.

- **Overrides.** If you do not want to use the default Oracle, SQL Server, or DB2 for z/OS JDBC driver(s), you can override the default driver values using the following settings:
  - **Database URL.** The database server URL.
  - **Database Driver.** The database driver. The driver JAR file(s) or a link to the file(s) must be in the `jlib` directory.

  Refer to the driver documentation for the URL and driver syntax.

  Example: If you want to use a DataDirect SQL Server driver, you might enter the following:

  Database URL. `jdbc:datadirect:sqlserver://serverabc.acmeco.corp:1433`

  Database Driver. `com.ddtek.jdbc.sqlserver.SQLServerDriver`

3 Click **Test** to confirm that the settings for the data source are correct. A message appears notifying you that the connection is successful or unsuccessful.

4 Click **OK** when you are finished. The CIMS Data Source entry appears in the CIMS Server JDBC Data Source Configuration dialog box.

5 Click **Apply** to apply the change to the `CIMSDBConfig.xml` file or click **OK** to apply the change and close the dialog box.

To set a CIMS Data Source as the default:

Click the CIMS Data Source that you want to set as the default, and then click **Set as Default**. `Yes` appears in the **Default** column for the data source.

To edit a CIMS Data Source:

1 Double-click the CIMS Data Source entry, or click the entry and then click **Edit**.

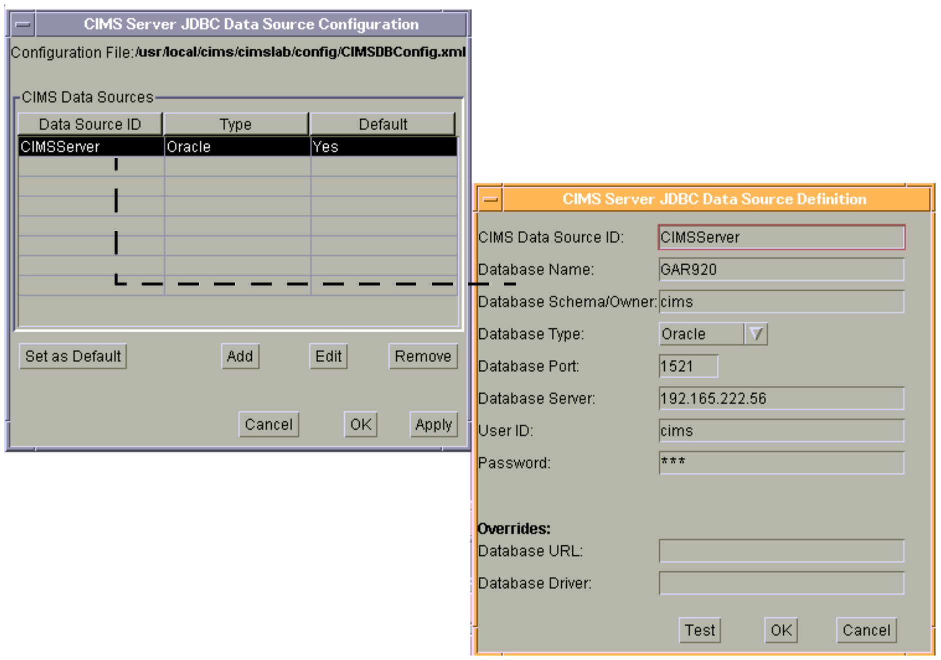
2 In the CIMS Server JDBC Data Source Definition dialog box, follow the steps in *To add a CIMS Data Source:* on page 1-14 beginning with Step 2.

To remove a CIMS Data Source:

Click the CIMS Data Source entry, and then click **Remove**.

Example of Creating CIMS Data Source Definition in CIMS Processing Engine Administrator

By default, CIMS Server for UNIX provides a CIMS Data Source named `CIMSServer` that points the Oracle database type. You can modify this data source to point to the database that you have created for CIMS Server for UNIX as shown in example Figure 1-2.


CIMS Server JDBC Data Source Configuration
Set as Default
Add
Edit
Remove
Cancel
OK
Apply
CIMS Server JDBC Data Source Definition
Test
OK
Cancel

Configuration File: /usr/local/cims/cimslab/config/CIMSDBConfig.xml

CIMS Data Sources

Data Source ID	Type	Default
CIMSServer	Oracle	Yes

CIMS Data Source ID:	CIMSServer
Database Name:	GAR920
Database Schema/Owner:	cims
Database Type:	Oracle
Database Port:	1521
Database Server:	192.165.222.56
User ID:	cims
Password:	***

Overrides:

Database URL:	
Database Driver:	

Figure 1-2 • Configuring CIMS Data Sources

Example of the Data Source Definition in the CIMSDBConfig.xml File

The `CIMSDBConfig.xml` file that contains the definition shown in Figure 1-2 would appear as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<CIMSDBConnections version= "1.0">
   <Connection
       dataSourceId="CIMSServer"
       dataBaseName="GAR920"
       dataBasePrefix="cims"
       dataBaseType="Oracle"
       dataBasePort="1521"
       dataBaseServer="192.165.222.56"
       user="cims"
       password="E3ah9XfNsMw=">
   </Connection>
</CIMSDBConnections>
```

Using the CIMSDBConfig Utility to Modify the CIMSDBConfig.xml File

USS on z/OS does not provide an XTERM server. You can rlogin into USS from another UNIX server and set `DISPLAY` to use the base system's XTERM server or you can use the CIMSDBConfig utility, which is a command line utility that updates the `CIMSDBConfig.xml` file.

The syntax for the CIMSDBConfig utility is:

```
java -jar CIMSDBConfig.jar [mode] [parameters]
```

Where:

`mode =`	
`?`	`Display this hlep`
`-add`	`Add a new connection`
`-delete`	`Delete a connection`
`-update`	`Update a connection`
`-default`	`Set the connection as the default`
`-test`	`Test the connection`
`parameters =`	**Note:** For more detailed descriptions of the following parameters,see *To add a CIMS Data Source:* on page 1-14. The parameters are the same as those in the CIMS Processing Engine Administrator GUI.
`-d datasourcename`	`The data source name`
`-n databasename`	`The database name (location DB2)`
`-o owner`	`The database owner or schema`
`-p port`	`The database port`
`-s server`	`The server where the database resides`
`-t type`	`DB2, Oracle, or SQL Server`
`-u user`	`Database user for sign on`
`-w`	`User password for sign on`
`-r url`	`URL override`
`-v driver`	`Database driver override`

Example of Creating CIMS Data Source Definition in the CIMSDBConfig.xml File Using the CIMSDBConfig Utlitity

The CIMS Data Source Definition example on page 1-17, which was produced using the CIMS Processing Engine Administrator GUI, could have also been produced by executing the CIMSDBConfig utility twice. The first execution would add the data source and the second execution would make the data source the default as shown in the following examples.

To add the data source:

```
Java -jar CIMSDBConfig.jar -add -d CIMSServer -n GAR920 -o cims -t Oracle -p
1521 -s 192.165.222.56 -u cims -w CIMS123
```

To make the data source the default:

```
Java -jar  CIMSDBConfig.jar -default -d CIMSServer
```

Modifying the cims.par File

The `$CIMS_HOME/cims.par` file contains the parameters needed to run CIMS Processing Engine. This file contains the following parameters. In most cases, you will not need to change this file other than to enter the license key.

- **defaultDataSourceID.** This is the default CIMS Data Source ID to be used by CIMS Server for UNIX. This value is set using the CIMS Processing Engine Administrator or CIMSDBConfig utility.

  You can override the default CIMS Data Source ID if needed by including the `dataSourceId` parameter in the XML control file. For more information about this file, see *Setting Up the Control File (ProcCntl.xml)* on page 3-15.

- **LicenseKey.** CIMS Server for UNIX requires a license key that is provided by CIMS Lab. You can enter the license key manually, or if you have access to CIMS Processing Engine Administrator, you can click **License** in the CIMS Server JDBC Data Source Configuration dialog box and type in the license.

  If you do not have your license key, contact CIMS Lab.

- **Javapath.** The directory where the JRE is installed. This value is populated automatically when CIMS Server for UNIX is installed.

- **Processesdir.** The directory that contains the process definitions. The default directory is `$CIMS_HOME/processes`. For more information about process definitions, see *Process Definitions (processes Directory)* on page 2-6.

Example cims.par File

```
defaultDataSourceId=CIMSServer
LicenseKey=999999999999
javapath=/opt/java1.4/bin
processesdir=/usr/local/cims/cimslab/processes
```

Set Up CIMS Server for UNIX

The administrative tasks required to set up CIMS Server for UNIX are performed using the CIMS Server Administrator application. Refer to the following tasks in the Getting Started section in Chapter 1 of the *CIMS Server Administrator's Guide* to start the set up and then refer to the other chapters in that guide for more detailed information.

- Create a CIMS Data Source that enables the CIMS Server Administrator to access the database. This is a different procedure than described in *Modifying the CIMSDBConfig.xml File* on page 1-14.
- Select and initialize the database.
- Enter the configuration and administration settings for CIMS Server for UNIX.

  Note that the process definition path in the CIMS Server Administrator Configuration dialog box and tasks described in the *Processing Data* chapter and other chapters of the *CIMS Server Administrator's Guide* are not applicable if you are using the CIMS Server for UNIX version of CIMS Processing Engine. The CIMS Server for UNIX Processing Engine is run using the files and scripts described in *Chapter 3, Processing Data* of this guide. At this time, there is no GUI interface for the CIMS Server for UNIX version of CIMS Processing Engine.

Run a Sample Process

CIMS Server for UNIX includes six default process definition directories in the `$CIMS_HOME/processes` directory: `UnixOS`, `UnixFS`, `UnixORA`, `UnixORAstorage`, `UnixDB2`, `UnixDB2storage`, and `Apache`. Each of these directories contains sample CSR files. To process these CSR files and load the output data into the database, run the `$CIMS_HOME_nightly.sh` script. (For information about the `nightly.sh` script, see page 2-11. For information about the `processes` directory and process definition subdirectories, see page 2-6.)

To verify that the output data has been loaded to the database, start CIMS Server Administrator and click **Chargeback Administration ▸ Database Loading ▸ Load Tracking**. The data for each process definition appear as Summary, Ident, and Detail files on the **Loads** tab. For more information about these files, see *Chapter 3, Processing Data*.

2

System Architecture

This chapter describes the architecture of the CIMS Server for UNIX system.

CIMS Server for UNIX Architecture

The following is an overview of the components that comprise the CIMS Server for UNIX architecture.

The components are grouped by directory in `$CIMS_HOME`. Each directory contains the files needed to run CIMS Server for UNIX. It might be helpful to refer to the directories as you read the following sections.

Modifying CIMS Server for UNIX Files and Scripts

Important! • If you modify any of the files or scripts provided with CIMS Server for UNIX, it is very important that you rename the file. Otherwise, the file will be overwritten when you upgrade to a new version of CIMS Server for UNIX.

Although you can modify any of the files and scripts shipped with CIMS Server for UNIX, the following scripts will most likely require modification:

- `$CIMS_HOME/scripts/nightly.sh`
- `$CIMS_HOME/scripts/nightly.pl`
- `$CIMS_HOME/scripts/loadCIMS.sh`
- `$CIMS_HOME/scripts/loadCIMS.pl`
- `$CIMS_HOME/processes/`*`process definition`*`/Job`*`definitionname`*`.pl`
- `$CIMS_HOME/processes/`*`process definition`*`/ProcCntl.xml`

Instructions for modifying these files and scripts are included in this guide. All CIMS Server for UNIX scripts also include instructions in the script.

Collection Files (collectors Directory)

There are two types of collectors that are used with CIMS Server for UNIX: external collectors and internal collectors.

- External collectors are CIMS Data Collectors for Windows, Mainframe, and UNIX. These collectors collect usage metering data and produce CIMS Server Resource (CSR) files or CIMS Server Resource Plus (CSR+) files, which can be sent to CIMS Server for UNIX for processing. External collectors are described in detail in the CIMS documentation for each collector (see *Related Publications* on page ix).
- Internal collectors are internal to CIMS Server for UNIX. These collectors collect and convert the data from usage metering files to produce CSR files. These CSR files can be processed by CIMS Server for UNIX, CIMS Server, or CIMS Mainframe.

The `$CIMS_HOME/collectors` directory contains a subdirectory for each of the internal collectors provided with CIMS Server for UNIX. Each subdirectory contains a conversion script that is used to collect data and produce the CSR files. The conversion script is described *Conversion Script* on page 2-3.

In addition to those internal collectors provided with CIMS Server for UNIX by default, you can also add internal collectors as needed. To quickly create a subdirectory and conversion script for a collector, copy and rename the `collectors/Universal` subdirectory and then rename and modify the `Universal.pl` script within the subdirectory.

Each internal collector subdirectory contains one or more feed subdirectories that contain the usage metering files to be processed. The feed subdirectories designate the source of the files.

Important! • To prevent data processing errors, the subdirectory for each of the internal collectors should not contain subdirectories other than feed directories and feed directories should not contain files other than usage metering files.

Conversion Script

Many collectors use a conversion script, *`collectorname`*`.pl`, to convert usage metering files to CSR files. The conversion script performs conversion and processing tasks including the following:

- Reads the usage metering files contained in the feed subdirectory for the collector (i.e., `$CIMS_HOME/collectors/<`*`collector name`*`>/<`*`feed`*`>`). The files are named `<Date>.txt`, where `Date` is in yyyymmdd format. If there are multiple feed subdirectories for a collector, the conversion script will search for usage metering files in all subdirectories.
- Calls CIMS Perl Aggregation Engine (if applicable). CIMS Perl Aggregation Engine (`cs_agtools.pm`) is a Perl module that aggregates the records within a usage metering file by identifier values. That is, if multiple records within a file contain the same identifier values, CIMS Aggregation Engine will produce one record that contains sum total resource values for the rate codes within these records. Aggregation reduces the amount of data that CIMS Processing Engine must process and improves processing time.
- Defines the chargeback identifiers and resources that are collected from the usage metering data for input into the CSR file. (Note that this is not applicable to all collectors.)

  CIMS Lab defines the most useful identifiers and resources for each collector in the collector's conversion script. These are the identifiers and resources that appear in the CSR file records.

  For many collectors, CIMS Lab pre-loads the resources defined in the conversion script as rate codes in CIMSRate table. You can then use CIMS Server Administrator to modify the options for these rate codes, such as description and monetary value, for your site. However, the rate codes for some collectors are not pre-loaded in the CIMSRate table and must be added as described in the *CIMS Server Administrator's Guide*.

  If you want to define identifiers and/or resources other than the default values in the conversion script, you need to modify the script. Note that if you want to use resources other than those defined, you need to add the rate codes for any new resources to the CIMSRate table.

- Calls the `Cleanup` subroutine to purge usage metering files from the feed subdirectories. Usage metering files older the than `Clean_Age` parameter value will be purged. For more information about the `Cleanup` subroutine, see *About the Scan and Cleanup Subroutines* on page 2-7.

Conversion Script Parameters

The conversion scripts for all collectors require the parameters shown in the following table.

Parameter	Description/Values
`LogDate`	The log date specifies the date for the data that you want to collect. For more information about using a log date, including valid log date values, see *nightly.sh* on page 2-11.
`RetentionFlag`	This parameter is for future use.
`AllowMissingFiles`	Specifies whether a warning or error occurs when feed subdirectories do not contain a file that matches the log date value. Valid values are: ■ "`TRUE`" (a warning occurs, processing continues) ■ "`FALSE`" (an error occurs, processing fails)
`AllowEmptyFiles`	Specifies whether a warning or error occurs when feed subdirectories contain a zero-length file that matches the log date value. Valid values are: ■ "`TRUE`" (a warning occurs, processing continues) ■ "`FALSE`" (an error occurs, processing fails)
`OutputFolder`	The process definition directory for the collector.
`LogFolder`	The collector subdirectory that contains the conversion script. For example, if you are running the Apache conversion script (`Apache.pl`) the subdirectory is `/usr/cims/collectors/Apache`. The conversion script will search for all usage metering files in the collector's feed subdirectories that match the `LogDate` parameter value.
`LogFileName (optional)`	By default, the conversion script searches for usage metering files that match the `LogDate` parameter value. However, you can use this parameter to specify another file name.

Table 2-1 • Conversion Script Parameters

Configuration Files (config) Directory

The `$CIMS_HOME/config` directory contains the following files:

`CIMSDBConfig.xml`	This file contains the parameters needed to connect to one or more databases within your organization. This file is described in *Modifying the CIMSDBConfig.xml File* on page 1-14.
`cims.par`	This file contains the parameters needed to run CIMS Processing Engine. This file is described in *Modifying the cims.par File* on page 1-20.

CIMS Processing Engine Programs (jlib Directory)

CIMS Processing Engine is a fast, efficient data processing engine capable of handling large volumes of data. CIMS Processing Engine consists of the following programs. Each of these programs is a JAR file in the `$CIMS_HOME/jlib` directory.

- **CIMSAcct.** This program processes the CSR or CSR+ files provided by the external or internal collectors (see *Collection Files (collectors Directory)* on page 2-2) and performs functions such as account code conversion. CIMSAcct produces a CSR+ output file, which is sorted by the CIMSSort subroutine and sent to CIMSBill. For more information about the CIMSSort subroutine, see *About the Scan and Cleanup Subroutines* on page 2-7.
- **CIMSBill.** This program processes the sorted CSR+ file from CIMSAcct and builds output files that contain the billing information that is used to generate invoices and reports.
- **CIMSLoad.** This program loads the output files from CIMSAcct and CIMSBill into an Oracle, SQL Server, or DB2 for z/OS database.

For more information about the CIMS Processing Engine components, see *Chapter 3, Processing Data*.

CIMS Perl Modules (CIMS_lib Directory)

This directory contains the following Perl modules used by the CIMS Server for UNIX job and conversion scripts:

`$CIMS_HOME/lib/CIMS_lib/CS_libs.pm`	This module contains subroutines that validate arguments, search directories, scan files, and perform other operational tasks required by job and conversion scripts.
`$CIMS_HOME/lib/CIMS_lib/CS_aggtools.pm`	This module contains the subroutines and objects used to support the CIMS Perl Aggregation Engine.

Process Definitions (processes Directory)

The CIMS Server for UNIX installation includes the `$CIMS_HOME/sample_processes` directory. The subdirectories in `sample_processes` are referred to as process definitions. A process definition is a directory that contains the files required to process usage data from a particular source such as a database, operating system, or application.

When you install CIMS Server for UNIX, the `sample_processes` directory and all of its contents are copied to the `$CIMS_HOME/processes` directory and the `processes` directory is defined in the `cims.par` configuration file. (See *Modifying the cims.par File* on page 1-20.) Each time that you re-install CIMS Server for UNIX or upgrade to a new release, the application searches for the `processes` directory and recreates the directory if it does not exist.

When a new release of CIMS Server for UNIX contains a new process definition directory or changes to an existing subdirectory or its contents, the additions and changes are made in the `sample_processes` directory. You can then copy the new or updated directories/files to the `processes` directory.

To help to ensure that you do not lose data, CIMS Lab recommends that you include the `processes` directory in your back up schedule.

Each process definition directory contains the following. Each of these components are described in the following sections.

- A Perl job script, `Job`*`definitionname`*`.pl`, that contains the processing instructions for CIMS Processing Engine.
- One or more feed subdirectories that contain the CSR or CSR+ files created by the external or internal CIMS Data Collectors.
- Files used to process the data in the CSR or CSR+ files, including the `ProcCntl.xml` file. This file contains the parameters for the CIMSAcct, CIMSBill, and CIMSLoad programs.

A separate process definition is required for each subsystem that you collect data from. CIMS Lab provides default process definition directories for the CSR files produced by CIMS Data Collector for UNIX and the CIMS Server for UNIX internal collectors.

If a process definition directory does not exist for the collector, you can simply copy and rename an existing directory and modify the job script.

Job Script

The Perl job script, `Job`*`definitionname`*`.pl`, performs the following tasks depending on whether you are using CIMS Server for UNIX to process data from external or internal data collectors.

If you are using CIMS Server for UNIX to process CSR or CSR+ files produced by *external* collectors, the job script performs the following tasks:

1. Calls the `Scan` subroutine to gather the *`yyyymmdd`*`.txt` CSR or CSR+ files in the process definition subdirectories, where the date is in yyyymmdd format. The files that are gathered depend on the `LogDate` parameter that you supply (see *Using the LogDate Parameter*). For example, if you provide a `LogDate` parameter of `20050318`, all files named `20050318.txt` are gathered.
2. Calls the `procCIMS.pl` script to run CIMSAcct, CIMSBill, and CIMSLoad (see *procCIMS.pl* on page 2-13).
3. Calls the `Cleanup` subroutine to purge files from the process definition directory and feed subdirectories.

If you are using CIMS Server for UNIX to process CSR or CSR+ files produced by the internal collection process, the job script first calls the conversion script before performing the preceding steps.

Using the LogDate Parameter

The job script requires the `LogDate` parameter to select the files that are processed by CIMS Server for UNIX. This parameter is entered in the command line when the `nightly.sh` script is run. For more information about the `LogDate` parameter and the `nightly.sh` script, see *nightly.sh* on page 2-11.

About the Scan and Cleanup Subroutines

Note • The Scan and Cleanup subroutines are in the Perl module `CS_libs.pm` (see *CIMS Perl Modules (CIMS_lib Directory)* on page 2-5).

The job script calls the Scan subroutine to perform the following tasks:

1. Verifies that the subdirectory or subdirectories in the process definition directory contain a CSR or CSR+ file with a date in its file name that matches the `LogDate` parameter.
2. If there are multiple subdirectories that contain files that match the `LogDate` parameter, concatenates the CSR or CSR+ files from each subdirectory into one file.
3. Outputs the final CSR or CSR+ file to the process definition directory. The file name for the final CSR or CSR+ file is either *`yyyymmdd`*`.txt` or `CurrentCSR.txt` depending on the value that you set for the `retainDateFlag` parameter in the job script.

The job script calls the `Cleanup` subroutine to delete CSR or CSR+ files that have yyyymmdd in the file name from the process definition subdirectories. The job script includes a `Clean_Age` parameter that enables you to specify the number of days that you want to retain files before they are deleted.

Modifying the Job Script

The job script calls and passes the following parameters to the conversion script (if applicable), the `Scan` and `Cleanup` subroutines, and the `procCIMS.pl` script. You need to modify these parameters for you organization as needed.

Parameter	Description/Values
`this_job`	The name of the process definition directory in which the job script resides. Do not change this parameter unless the name of the process definition directory has been changed.
`job_xml`	The XML file that contains the parameters for the CIMSAcct, CIMSBill, and CIMSLoad programs. The default file name is `ProcCntl.xml`. Do not change this parameter unless you have changed the file name. For more information about the `ProcCntl.xml` file, see *Setting Up the Control File (ProcCntl.xml)* on page 3-15.
`retainDateFlag`	Specifies whether the date is retained in the CSR or CSR+ file that is produced by the Scan subroutine (i.e., *yyyymmdd*`.txt` rather than `CurrentCSR.txt`). Valid values are: ■ `"TRUE"` (the file name is *yyyymmdd*`.txt`) ■ `"TRUE"` (the file name is `CurrentCSR.txt`) The default is `"TRUE"`.
`allowEmptyFiles`	Specifies whether a warning or error occurs if the feed subdirectories contain a zero-length file that match the log date value. Valid values are: ■ `"TRUE"` (a warning occurs, processing continues) ■ `"FALSE"` (an error occurs, processing fails) The default is `"TRUE"`.
`allowMissingFiles`	Specifies whether a warning or error occurs when feed subdirectories do not contain a file that matches the log date value. Valid values are: ■ `"TRUE"` (a warning occurs, processing continues) ■ `"FALSE"` (an error occurs, processing fails) The default is `"TRUE"`.

Table 2-2 • Parameters Passed From the Job Script

Parameter	Description/Values
`Clean_Age`	The number of days that you want to keep *yyyymmdd*`.txt` CSR or CSR+ files in the process definition directory after their creation date. **Example:** `Clean_Age="60"` This example specifies that all files that are older than 60 days from the current date are deleted. The default is 45 days from the current date. **Note:** CSR or CSR+ files named *yyyymmdd*`.txt` appear in the process definition directory only when the `retainDateFlag` parameter is set to `TRUE`. Otherwise, the files are named `CurrentCSR.txt`. In this case, the existing CSR/CSR+ file is overwritten each time a new file is created. Therefore, excess files are not accumulated and file clean up is not required.
`SubDirs`	Specifies whether the *yyyymmdd*`.txt` files that are contained in the feed subdirectories are deleted. The files are deleted on the schedule set by the `Clean_Age` parameter. Valid values are: ■ `"TRUE"` (the files are deleted) ■ `"FALSE"` (the files are not deleted) The default is `"TRUE"`.

Table 2-2 • Parameters Passed From the Job Script (Continued)

Feed Subdirectory

Feed subdirectories are used to store CSR or CSR+ files from external and internal collectors. The CSR or CSR+ file name contains a date in yyyymmdd format.

The Scan subroutine processes and concatenates the CSR or CSR+ files in the feed subdirectories by the date in the file name. The resulting output file is placed directly in the process definition directory. For more information, see *About the Scan and Cleanup Subroutines* on page 2-7.

Important! • To prevent data processing errors, the process definition directory should not contain subdirectories other than feed subdirectories and feed subdirectories should not contain files other than CSR or CSR+ files.

Job Log Files (log Directory)

The `$CIMS_HOME/log` directory contains log files that contain the results of each job that you ran by day. If a warning or failure occurs during processing, the file indicates at which point the warning/failure occurred.

The contents of the job file are included in an e-mail message that notifies recipients of successful, successful with warning, or failed processing. The e-mail recipients are defined in the `nightly.sh` script as described on page 2-11.

Note • A job log file is not created until the job is run. If an error occurs and the job is not run (for example, the job script contains a syntax error) a log file is not generated and a notification e-mail is not sent. To ensure that the job runs correctly and that a log file is generated, you can run the job script from the command line.

The log files are named `nightly_PREDAY_`*`yyyymmdd`*`.txt` or `nightly_`*`yyyymmdd`*`.txt`, where `nightly_`*`yyyymmdd`*`.txt` is the most recent log. The date in `nightly_PREDAY_`*`yyyymmdd`*`.txt` is the day that the `nightly.sh` script was run. The date in `nightly_`*`yyyymmdd`*`.txt` is the date specified by the `LogDate` parameter.

Scripts (scripts Directory)

The `scripts` directory contains the scripts described in the following sections. The scripts are ordered by their position in the processing cycle.

nightly.sh

The CIMS Server for UNIX installation includes the `$CIMS_HOME/scripts/sample_nightly.sh` script. When you install CIMS Server for UNIX, the `$CIMS_HOME/scripts/sample_nightly.sh` script is copied to create the `scripts/nightly.sh` script. Each time that you re-install CIMS Server for UNIX or upgrade to a new release, the application searches for the `nightly.sh` script and recreates the script if it does not exist.

You need to set variable values in the `nightly.sh` script. If a new release of CIMS Server for UNIX contains changes to the `sample_nightly.sh` script, you can incorporate those changes into the `nightly.sh` script that you have modified rather than replacing the script. If you replace the script, any existing variable values that you have set will be overwritten.

The `nightly.sh` script runs CIMS Server for UNIX by performing the following tasks:

- Calls the `nightly.pl` Perl script to begin processing.
- Calls the mailx utility to send e-mail notification of successful, successful with warning, or failed processing.

The `nightly.sh` script must be scheduled to run on a regular basis. You can use any batch scheduler to run this script. CIMS Lab recommends that you run the script using the CIMS user account designated during the installation of CIMS Server for UNIX.

Example

To schedule the `nightly.sh` script to run every morning at 4:15, the following crontab entry could be used:

```
15 4 * * * /usr/cims/scripts/nightly.sh
```

Defining Variables in the nightly.sh Script

You need to define the following variables in the `nightly.sh` script:

`MAIL_UTIL`	The name of the mail utility used to send mail messages on this computer. Usually mail or mailx.
`MAIL_LIST`	A comma-separated list of e-mail recipients. For example, `csanchez@xyzco.com,bhughes@xyzco.com,....`

Using the LogDate Parameter

The `nightly.sh` script requires the `LogDate` parameter. This parameter is used to select the files that are processed by CIMS Server for UNIX.

If you are using CIMS Server for UNIX to process CSR or CSR+ files from external collectors, this parameter is used to select the CSR or CSR+ files that contained in the feed subdirectories.

If you are using CIMS Server for UNIX to process CSR or CSR+ files from internal collectors, this parameter is used to select the usage metering files that are then processed and converted to CSR+ files.

The following are valid values for the `LogDate` parameter.

- `preday` (previous day)
- `rndate` (current day)
- `curday` (current day and previous day)
- date in `yyyymmdd` format

This parameter is entered in the command line when running the `nightly.sh` script.

nightly.pl

Note • The `nightly.pl` script follows the same conventions as described for the `nightly.sh` script (see *nightly.sh* on page 2-11).

The `nightly.pl` script calls all jobs that you want to run. This script contains the Perl array `@job_list`, which defines the process definition jobs that will be executed. Jobs are called in the order that they appear in the list. You can add and remove jobs to and from this list.

The `nightly.pl` script requires the `LogDate` parameter, which is entered in the command line when running the `nightly.sh` script.

The `nightly.pl` script contains the `stopOnProcessFailure` parameter. If set to `TRUE`, the nightly process will stop if a failure is returned for a job. If set to `FALSE`, processing will continue to the next job defined in the `@job_list` array.

loadCIMS.sh and loadCIMS.pl

The loadCIMS.sh and loadCIMS.pl scripts support the CIMS Mainframe Data Collectors and CIMS Server for DB2. For more information about these scripts, refer to the *CIMS Server for DB2 Database Setup Guide*.

procCIMS.pl

The `procCIMS.pl` script calls the CIMSAcct, CIMSBill, and CIMSLoad programs described in *CIMS Processing Engine Programs (jlib Directory)* on page 2-5.

The `procCIMS.pl` script also calls the CIMSSort subroutine that produces a sorted version of the CSR+ Output file created by CIMSAcct. This file is used as input into CIMSBill.

The `procCIMS.pl` script requires the parameters shown in the following table. These parameters are passed from the job script.

Parameter	Description/Values
`ProcessName`	The process definition directory that contains the XML control file for CIMSAcct, CIMSBill, and CIMSLoad.
`XML_JobFile`	The control file for CIMSAcct, CIMSBill, and CIMSLoad. The default file name is `ProcCntl.xml`. This file contains the parameters used by the processing programs.

Table 2-3 • procCIMS Script Parameters

3

Processing Data

This chapter provides an overview of the CIMS Server for UNIX data processing cycle and components.

About Processing Data

CIMS Server for UNIX processes and applies business rules to resource usage data from any application, system, or operating system on any platform. The primary method for loading this data into CIMS Server is the CIMS Server Resource (CSR) or CIMS Server Resource Plus (CSR+) file. The file type depends on the collector that created the file (see *About CSR and CSR+ Files* on page 1-4).

The CIMS Data Collector(s) that you are using to collect your system data automate the data processing cycle. The collectors convert usage metering data created by your system into CSR or CSR+ files. Once the CSR or CSR+ files are created, CIMS Processing Engine processes the data in the files and loads the resulting output files into the database using the components `CIMSAcct.jar`, `CIMSBill.jar`, `CIMSLoad.jar`, and the CIMSSort subroutine.

Data Processing Frequency

The preferred method of processing is to run the full data processing cycle as the data becomes available. The data produced by the various operating systems (Mainframe, UNIX, Windows, etc.) and applications/databases (CICS, DB2, Oracle, IIS, Exchange Server, etc.) are usually made available for processing on a daily basis. Other feeds such as time accounting, help desk, line charges, equipment charges, and other shared services are usually produced on a monthly basis.

There are several advantages to running the full costing cycle on a daily or data availability basis:

- The volume of data created makes it more practical to process daily. For example, a daily Apache Web server access log might contain millions of records. It is more efficient to process these records each day of the month rather than try to run many millions of records through the processing cycle at month end.
- It is easier to catch processing errors when the data is reviewed on a daily basis. It is more difficult to troubleshoot a problem when it is discovered at month end. If an unusual increase in utilization is observed for a specific resource at month end, the entire month’s records must be checked to determine when the increase first took place.

  Because there are fewer jobs, transactions, or records to review, the task of determining what caused the utilization spike is much simpler if caught on the day in which it occurred.
- If the program CIMSBill is run monthly, the start date is the first day of the month and the end date is the last day of the month. Because of this date range, it is not possible to view CIMS Summary records for a single day or week. The smallest time range that may be used is the entire month.

Required Directory Permissions for Data Processing

The administrator that executes processing using CIMS Server for UNIX requires full access to files in the `processes` directory (that is, the ability to create, modify, delete, overwrite, etc.). Therefore, the UNIX user and group for the CIMS Server for UNIX administrator must have read, write, and execute permissions (mod 0770) for the `processes` directory and all subdirectories.

CIMS Processing Engine Components

CIMS Processing Engine is composed of Java programs and Perl modules. Each engine component performs particular functions in the processing cycle. The following table describes the components in the order that they appear in the processing cycle and the output files created by the objects as applicable.

Component	Description	Output Files
`CIMSAcct.jar`	Performs account code conversion, shift determination, and date selection on the data contained in the CSR or CSR+ file. For more information about CIMSAcct, including a description of its input and output files, see page 3-6.	■ **CIMSAcct Output CSR+.** This file is input into the CIMSSort subroutine or can be reprocessed through CIMSAcct.
CIMSSort subroutine	Sorts the CIMSAcct Output CSR+ file and produces a version of the file that is ready to be processed by CIMSBill.	■ **Sorted CIMSAcct Output CSR+.** This file is input into CIMSBill.
`CIMSBill.jar`	Processes the sorted CIMSAcct Output CSR+ file and performs shift processing, CPU normalization, and include/exclude processing and creates files that contain the billing information used to generate invoices and reports. For more information about CIMSBill, including a description of its input and output files, see page 3-10.	■ **CIMSBill Detail.** This file is loaded into the database or can be reprocessed through CIMSBill. ■ **CIMS Summary.** This file is loaded into the database or can be reprocessed through CIMSAcct . ■ **CIMS Ident.** This file is loaded into the database.
`CIMSLoad.jar`	Loads the CIMSBill Detail, CIMS Summary, and CIMS Ident files into the database. For more information about CIMSLoad, including a description of its input and output files, see page 3-13.	

Table 3-1 • CIMS Processing Engine Components

Component	Description	Output Files
`CS_libs.pm`	Contains Perl subroutines that support the job and processing scripts. For more information about this module, see page 3-14.	
`CS_aggtools.pm`	Contains Perl subroutines that support the CIMS Perl Aggregation Engine. For more information about this module, see page 3-14.	

Table 3-1 • CIMS Processing Engine Components (Continued)

CIMSAcct

CIMSAcct is the first data processing program that is executed. CIMSAcct processes the records in the input file (see *CIMSAcct Input* on page 3-7) and performs the following:

- **Account code conversion.** CIMSAcct uses specified identifiers from the input file to build the account code and perform account code conversion if required. For more information, see *Setting Up Account Codes and Performing Account Code Conversion* on page 3-28.
- **Date selection.** CIMSAcct selects the input file records to be processed based on the specified date or date range.
- **Shift determination.** CIMSAcct can determine the shift to be used for processing in either of the following ways:
    - Use the shift code from the input file records. If a shift code is not included in the records, the default shift code is 1.
    - Recalculate the shift using the start date/time in the records. For more information, see *Setting Up Shifts* on page 3-38.

CIMSAcct produces the Output CSR+ file, which contains records that are properly formatted for input into CIMSBill.

Reprocessing the CIMSAcct Output CSR+ Files Through CIMSAcct

You can reprocess the CIMSAcct Output CSR+ file through CIMSAcct. This is usually done to reprocess data for further account code conversion. To reprocess this file, you need modify the `procCIMS.pl` script. For more information about this script, see page 2-13.

CIMSAcct Input

The following table lists the input and processing files used by CIMSAcct. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
Input Files (CIMSAcct can process usage data from any of the following sources)		
CSR or CSR+ file	`CurrentCSR.txt` **Or** *`yyyymmdd`*`.txt`	These are the most commonly used files for processing data through CIMS Server for UNIX. To view the record layout for these files, refer to the *CIMS Server Administrator's Guide*.
CIMSAcct Output CSR+ file	`AcctCSR.txt`	This file is the output from a previous run of CIMSAcct. This file provides an account code for each input file record. This file is input into the CIMSSort subroutine. When used as input into CIMSAcct, this file is used primarily for further account code conversion.
Processing Files		
Control file	`ProcCntl.xml` or `AcctCntl.xml`	These files contain the CIMSAcct parameters. `ProcCntl.xml` contains the parameters for CIMSAcct, CIMSBill, and CIMSLoad. `AcctCntl.xml` contains the parameters for CIMSAcct only. Unless you need to use separate control files (for example, you want to run CIMSAcct twice for additional account code conversion), CIMS Lab recommends that you use the `ProcCntl.xml` file.
Account Code Conversion table	`AcctTabl.txt`	This optional file contains the account code conversion table (see *Creating an Account Code Conversion Table* on page 3-33).

Table 3-2 • CIMSAcct Input

File Type	File Name	Description
Data Source Configuration file	`CIMSDBConfig.xml`	This file contains the database connection parameters. Database access is required by CIMSAcct only if the `DATE SELECTION PREVIOUS` or `DATE SELECTION CURRENT` control statement is used. Because the period is defined in the CIMSCalendar table in the database, CIMSAcct requires access to the table to determine the period.
CIMS Parameter file	`cims.par`	This file contains the parameters used to run CIMSAcct and the other CIMS Processing Engine components.

Table 3-2 • CIMSAcct Input (Continued)

CIMSAcct Output

The following table lists the CIMSAcct output files. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
CIMSAcct Output CSR+ file	`AcctCSR.txt`	This file provides an account code for each input file record. This file is input into the CIMSSort subroutine.
CIMSAcct Message file	User Defined	This optional file contains processing messages and results related to CIMSAcct. If a message file is not defined in the control file (see page 3-18), all CIMSAcct messages will go to STDOUT.
Exception file	`Exception.txt`	This is an optional file in CSR or CSR+ format that contains all records that did not match during account code conversion. These files can be reprocessed through the processing cycle as described in *About Exception File Processing* on page 3-27. To produce this file, you must have exception file processing enabled (see page 3-27).

Table 3-3 • CIMSAcct Output

CIMSSort

The CIMSSort is a Perl subroutine that sorts the CIMSAcct Output CSR+ file by account code and produces a version of the file that is ready to be processed by CIMSBill.

CIMSBill

The cost of information services, and which departments are using the services, is of considerable interest to an organization. When users are made aware of the costs and are held financially responsible for those costs, they are more likely to use the resources in a prudent manner. CIMS Server provides comprehensive computer center billing through the program CIMSBill.

The primary function of CIMSBill is to perform cost extensions within CIMS Server and to summarize cost and resource utilization by account code. CIMSBill uses the rate code table assigned to the client to determine the amount to be charged for each resource consumed. It is possible to have a unique rate code table for each client.

There are specific rules that CIMSBill follows when charging resource consumption:

- If a resource has a corresponding entry with a monetary amount in the rate code table, the number of resource units are multiplied by this amount. The result appears in the CIMS Summary records.
- If a resource has a corresponding entry with no or zero currency amount in the rate code table, a zero cost appears in the CIMS Summary records for that resource.
- If a resource *does not* have a corresponding entry in the rate code table, the data for that resource does not appear in the CIMS Summary records.
- If *none* of the resources in the CIMSBill input file have a corresponding entry in the rate code table, the CIMS Summary file will contain no records. You must have at least one rate code from the input file in the rate code table to produce CIMS Summary records.

In addition to standard costing, CIMSBill also performs the following:

- Processes non-standard costs such as miscellaneous, recurring, and credit transactions. For more information about creating and using transactions, refer to the *CIMS Server Web Reporting User's Guide*.
- Performs CPU Normalization. See *Normalizing CPU Values* on page 3-41.
- Performs include/exclude processing. See *Include/Exclude Processing* on page 3-43.

CIMSBill Input

The following table lists the input and processing files used by CIMSBill. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
Input File		
CIMSSORT/ CIMSAcct Output CSR+ file	`AcctCSR.txt`	This is the sorted Output CSR+ file from CIMSAcct.
Processing Files		
Control file	`ProcCntl.xml` or `BillCntl.xml`	These files contain the CIMSBill parameters. `ProcCntl.xml` contains the parameters for CIMSAcct, CIMSBill, and CIMSLoad. `BillCntl.xml` contains the parameters for CIMSBill only. Unless you want to use separate control files, CIMS Lab recommends that you use the `ProcCntl.xml` file.
Data Source Configuration file	`CIMSDBConfig.xml`	This file contains the database connection parameters.
CIMS Parameter file	`cims.par`	This file contains the parameters used to run CIMSBill and the other CIMS Processing Engine components.

Table 3-4 • CIMSBill Input

CIMSBill Output

The following table lists the CIMSBill output files. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
CIMSBill Resource file	`BillResource.txt` or a user defined name.	This optional file is the same as the CIMSBill Detail file except that no CPU normalization or include/exclude processing has been performed and accounting dates are not set. This file is produced only when the CIMSBill `resourceFile` attribute is provided (see page 3-20) and is loaded into the database only when the CIMSLoad `loadType` attribute is set to `Resource` (see page 3-23).
CIMSBill Detail file	`BillDetail.txt`	This file is loaded into the database for use in drill down reports. This file contains resource utilization data. The CIMSBill Detail file differs from the CIMSBill Resource file in that the Detail file reflects any CPU normalization or include/exclude processing that was performed on the CIMSAcct Output CSR+ file. The CIMSBill Detail file also includes accounting dates.
CIMS Summary file	`BillSummary.txt`	This file is loaded into the database for use in producing reports. This file contains both resource utilization and cost data.
CIMS Ident File	`Ident.txt`	This file contains all the identifiers that are contained in the input records (e.g., user ID, jobname, department code, server name). These identifiers are used during account code conversion to create your target account code structure. This file is loaded into the database.
CIMSBill Message file	User Defined	This optional file contains processing messages and results related to CIMSBill. If a message file is not defined in the control file (see page 3-21), all CIMSBill messages will go to STDOUT.

Table 3-5 • CIMSBill Output

CIMSLoad

CIMSLoad loads the output files from CIMSBill into the database. By default, CIMSLoad loads the CIMSBill Detail, CIMS Ident, and CIMS Summary files. However, you can use the `loadType` parameter to specify all files (including the CIMSBill Resource file) or just specific files as described on page 3-23.

CIMSLoad Input

The following table lists the input and processing files used by CIMSLoad. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
Input Files		
CIMSBill Resource file	`BillResource.txt` or a user defined name.	This optional file is the same as the CIMSBill Detail file except that no CPU normalization or include/exclude processing has been performed and accounting dates are not set. This file is loaded into the database only when the CIMSLoad `loadType` attribute is set to `Resource` (see page 3-23).
CIMSBill Detail file	`BillDetail.txt`	This file is loaded into the database for use in drill down reports. This file contains resource utilization data.
CIMS Summary file	`BillSummary.txt`	This file is loaded into the database for use in producing reports. This file contains both resource utilization and cost data.
CIMS Ident File	`Ident.txt`	This file contains all the identifiers that are contained in the input records (e.g., user ID, jobname, department code, server name). These identifiers are used during account code conversion to create your target account code structure. This file is loaded into the database.

Table 3-6 • CIMSLoad Input

File Type	File Name	Description
Processing Files		
Control file	`ProcCntl.xml` or `LoadCntl.xml`	These files contain the CIMSLoad parameters. `ProcCntl.xml` contains the parameters for CIMSAcct, CIMSBill, and CIMSLoad. `LoadCntl.xml` contains the parameters for CIMSLoad only. Unless you want to use separate control files, CIMS Lab recommends that you use the `ProcCntl.xml` file.
Data Source Configuration file	`CIMSDBConfig.xml`	This file contains the database connection parameters.
CIMS Parameter file	`cims.par`	This file contains the parameters used to run CIMSLoad and the other CIMS Processing Engine components.

Table 3-6 • CIMSLoad Input (Continued)

CIMSLoad Output

The following table lists the CIMSLoad output files. These files are in the process definition subdirectories in the `processes` directory.

File Type	File Name	Description
CIMSLoad Message file	User Defined	This optional file contains processing messages and results related to CIMSLoad. If a message file is not defined in the control file (see page 3-24), all CIMSLoad messages will go to STDOUT.

Table 3-7 • CIMSLoad Output

CS_libs

`$CIMS_HOME/lib/CIMS_lib/CS_libs.pm` is a Perl module that contains subroutines that validate arguments, search directories, scan files, and perform other operational tasks required by job and conversion scripts.

CS_aggtools

`$CIMS_HOME/lib/CIMS_lib/CS_aggtools.pm` is a Perl module that contains the subroutines and objects used to support the CIMS Perl Aggregation Engine.

Setting Up the Control File (ProcCntl.xml)

The parameters for the CIMSAcct, CIMSBill, and CIMSLoad programs are defined in an XML control file, `ProcCntl.xml`. A sample `ProcCntl.xml` file is included in each or the sample process definition directories provided with CIMS Server for UNIX. You can edit this file as needed.

CIMS Server for UNIX also includes separate XML control files for each program. These files are named `AcctCntl.xml`, `BillCntl.xml`, and `LoadCntl.xml` and contain the same parameters for that program as are contained in the `ProcCntl.xml` file. Unless you need to use separate control files (for example, you want to run CIMSAcct twice for additional account code conversion), CIMS Lab recommends that you use the `ProcCntl.xml` file.

Control File Example

For a description of the Step element and attributes, see page 3-16.

For descriptions of the parameters for the CIMSAcct program, see page 3-17.

For descriptions of all CIMSAcct and CIMSBill control statements, see page A-1.

For descriptions of the parameters for the CIMSBill program, see page 3-20.

For descriptions of the parameters for the CIMSLoad program, see page 3-22.

```
<?xml version="1.0" encoding="utf-8" ?>
<CIMSProcessing version= "4.0">
   <Step id="Process Acct"
      programName="CIMSAcct">
      <Parameters>
        <Parameter accCodeConvTable="AcctTabl.txt"/>
        <Parameter fullMessageFormat="True"/>
        <Parameter messageFile="AcctMsg.log"/>
        <Parameter trace="false"/>
        <Parameter ControlCard="Account Code Conversion Sort"/>
        <Parameter ControlCard="Print Account No-Match"/>
        <Parameter ControlCard="ACCOUNT FIELD0,SYSTEM_ID,1,8"/>
        <Parameter ControlCard="ACCOUNT FIELD1,OR_BASE,1,8"/>
        <Parameter ControlCard="DEFINE FIELD0,1,8"/>
        <Parameter ControlCard="DEFINE FIELD1,9,8"/>
        <Parameter ControlCard="DEFINE MOVEFLD0,1,8"/>
        <Parameter ControlCard="DEFINE MOVEFLD1,9,8"/>
     </Parameters>
    </Step>
   <Step id="Process Bill"
      programName="CIMSBill">
      <Parameters>
        <Parameter fullMessageFormat="true"/>
        <Parameter messageFile="BillMsg.log"/>
        <Parameter trace="false"/>
      </Parameters>
   </Step>
   <Step id="Load Detail"
     programName="CIMSLoad">
       <Parameters>
         <Parameter loadType="ALL"/>
         <Parameter bulkLoad="True"/>
         <Parameter detailFile="BillDetail.txt"/>
         <Parameter identFile="Ident.txt"/>
         <Parameter summaryFile="BillSummary.txt"/>
         <Parameter messageFile="LoadMsg.log"/>
         <Parameter trace="true"/>
       </Parameters>
    </Step>
</CIMSProcessing >
```

Control File Structure

This section describes the required and optional elements and attributes in a control file. Note that the sample control files provided with CIMS Server for UNIX do not include all of the attributes and parameters described in this section.

CIMSProcessing Element

The `CIMSProcessing` element is the root element of the control file. All other elements are child elements of `CIMSProcessing`. There are no attributes for the `CIMSProcessing` element.

Step Element

XML tree structure: `CIMSProcessing/Step`

A `Step` element defines a step within the control file. You can define one step for each program (CIMSAcct, CIMSBill, and CIMSLoad). If you define multiple steps for a program, only the attributes and parameters in first step are used.

The following table lists the attributes for the `Step` element.

Attribute	Required or Optional	Description
`id`	Required	A text string name for the step. **Example:** `id="Process Acct"` In this example, the step is executing the CIMSAcct program.
`programName`	Required	The name of the program that the step supports. Valid values are `CIMSAcct`, `CIMSBill`, or `CIMSLoad`.
`dataSourceId`	Optional	The CIMS Data Source ID to use as the default if other than the default specified in the `cims.par` file (see *Modifying the cims.par File* on page 1-20).

Table 3-8 • Step Element Attributes

Parameters Element

XML tree structure: `CIMSProcessing/Step/Parameters`

A `Parameters` element is a container for one or more `Parameter` elements.

Parameter Element

XML tree structure: `CIMSProcessing/Step/Parameters/Parameter`

A `Parameter` element defines a parameter to a step.

The following rules apply to parameter attributes:

- Some optional attributes have default values. If you do not include these attributes or provide blank values, the default values are used.
- For attributes that enable you to define the names of input and output files, do not include the path with the file name. These programs should reside in the process definition directory. The exception is the account code conversion table, which can be in any directory and requires a path if it is not in the process definition directory.

The following table lists the attributes for the Parameter element by program: CIMSAcct, CIMSBill, and CIMSLoad.

Program Name or Type	Attribute	Required or Optional	Description
Valid Parameters by Program Name			
`CIMSAcct` For a description of this program, see page 3-6. For an example of the parameters for this program in a job file see page 3-15.	`inputFile`	Optional	The name of the CSR or CSR+ file to be processed. This file must be in the process definition directory–do not include a path. The default is `"CurrentCSR.txt"`.
	`outputFile`	Optional	The name of the CIMSAcct Output CSR+ file. This file must be in the process definition directory—do not include a path. The default is `"AcctCSR.txt"`.

Table 3-9 • Parameter Element Attributes

Program Name or Type	Attribute	Required or Optional	Description
`CIMSAcct` (continued)	`accCodeConvTable`	Optional	The name of account code conversion table used by CIMSAcct. Include a path if the table is in a location other than the process definition directory. **Example:** `accCodeConvTable= "/usr/local/cims/cimslab/ AcctTabl.txt"` The default is `"AcctTabl.txt"`.
	`messageFile`	Optional	The name of the CIMSAcct Message and Results file. This file must be in the process definition directory—do not include a path. If this attribute is not provided or is left blank, all messages will go to Standard Output (STDOUT). The default is STDOUT.
	`exceptionFile`	Optional	The name of the exception file produced by CIMSAcct. This file must be in the process definition directory–do not include a path. The default is `"Exception.txt"`.
	`fullMessageFormat`	Optional	Specifies whether all messages produced by CIMSAcct are prefixed with the date and time. Valid values are: ■ `"true"` (the date and time are included) ■ `"false"` (the date and time are not included) The default is `"true"`.

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
`CIMSAcct` (continued)	`trace`	Optional	Specifies whether trace messages are produced in CIMSAcct Message and Results file or STDOUT. Valid values are: ■ `"true"` (trace messages are produced) ■ `"false"` (trace messages are not produced) The default is `"false"`.

	`ControlCard`	Optional	A valid CIMSAcct control statement.
			Example:
			`<parameter ControlCard= "PRINT ACCOUNT NO-MATCH" />`
			`<parameter ControlCard= "DEFINE FIELD0,1,8" />`
			For a description of all CIMSAcct control statements, see *CIMSAcct Control Statements* on page A-2.

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
`CIMSBill` For a description of this program, see page 3-10. For an example of the parameters for this program in a job file see page 3-15.	`inputFile`	Optional	The name of the CIMSAcct Output CSR+ file to be processed. This file must be in the process definition directory—do not include a path. The default is "`AcctCSR.txt`".
	`resourceFile`	Optional	The name of the optional CIMSBill Resource file. CIMS Lab recommends that you use the name `BillResource.txt`. This file must be in the process definition directory—do not include a path. If this attribute is not provided, the CIMSBill Resource file is *not* created.
	`detailFile`	Optional	The name of the CIMSBill Detail file produced. This file must be in the process definition directory—do not include a path. The default is "`BillDetail.txt`".
	`summaryFile`	Optional	The name of the CIMS Summary file produced. This file must be in the process definition directory—do not include a path. The default is "`BillSummary.txt`".

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
`CIMSBill` (continued)	`identFile`	Optional	The name of the CIMS Ident file. This file must be in the process definition directory—do not include a path. The default is "`Ident.txt`".
	`messageFile`	Optional	The name of the CIMSBill Message and Results file. This file must be in the process definition directory—do not include a path. If this attribute is not provided or is left blank, all messages will go to Standard Output (STDOUT). The default is STDOUT.
	`fullMessageFormat`	Optional	Specifies whether all messages produced by CIMSBill are prefixed with the date and time. Valid values are: ■ "`true`" (the date and time are included) ■ "`false`" (the date and time are not included) The default is "`true`".
	`trace`	Optional	Specifies whether trace messages are produced in CIMSBill Message and Results file or STDOUT. ■ "`true`" (trace messages are produced) ■ "`false`" (trace messages are not produced) The default is "`false`".

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description

`CIMSBill` (continued)	`ControlCard`	Optional	A valid CIMSBill control statement.
			Example:
			`<parameter ControlCard= "CLIENT SEARCH ON" />`
			`<parameter ControlCard= "DEFAULT CLOSE DAY=15" />`
			For a description of all CIMSBill control statements, see *CIMSBill Control Statements* on page A-11.

`CIMSLoad` For a description of this program, see page 3-13. For an example of the parameters for this program in a job file see page 3-15.	`bulkLoad`	Optional	Specifies whether to use bulk insertions or single record insertions into the database. Valid values are: ■ `"true"` (use bulk insertions) ■ `"false"` (use single record insertions) The default is `"true"`.
	`identFile`	Optional	The name of the CIMS Ident file to be loaded. This file must be in the process definition directory—do not include a path. The default is `Ident.txt`.
	`detailFile`	Optional	The name of the CIMSBill Detail file to be loaded. This file must be in the process definition directory—do not include a path. The default is `BillDetail.txt`.

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
`CIMSLoad` (continued)	`summaryFile`	Optional	The name of the CIMS Summary file to be loaded. This file must be in the process definition directory—do not include a path. The default is `BillSummary.txt`.
	`resourceFile`	Optional	The name of the optional CIMSBill Resource file. CIMS Lab recommends that you use the name `BillResource.txt`. This file must be in the process definition directory–do not include a path.
	`loadType`	Optional	Specifies the type of files to be loaded to the database. Valid values are: ■ `Summary` (CIMS Summary file) ■ `Detail`(CIMSBill Detail file) ■ `Resource` (CIMSBill Resource file) ■ `Ident` (CIMS Ident file) ■ `All` (All files except the CIMSBill Resource file) To specify files individually, you need to include a separate parameter for each file. The default is `All`.

Table 3-9 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
`CIMSLoad` (continued)	`messageFile`	Optional	The name of the CIMSLoad Message and Results file. This file must be in the process definition directory—do not include a path. If this attribute is not provided or is left blank, all messages will go to Standard Output (STDOUT). The default is STDOUT.
	`fullMessageFormat`	Optional	Specifies whether all messages produced by CIMSLoad are prefixed with the date and time. Valid values are: ■ `"true"` (the date and time are included) ■ `"false"` (the date and time are not included) The default is `"true"`.
	`trace`	Optional	Specifies whether trace messages are produced in CIMSLoad Message and Results file or STDOUT. Valid values are: ■ `"true"` (trace messages are produced) ■ `"false"` (trace messages are not produced) The default is `"false"`.

Table 3-9 • Parameter Element Attributes (Continued)

Accounting Dates

CIMS Server input records contain a *usage* start date and end date that specify the date range in which the resources in the record were consumed. CIMSBill uses the usage end date to calculate *accounting* start and end dates, which are used for billing and reporting. The accounting start and end dates may be the same as or different than the usage end date (see *How Accounting Dates Are Calculated* on page 3-26).

The accounting start and end dates are stored in the following fields in the CIMSBill output files:

- **CIMSBill Detail file.** The accounting dates are in the `ACCOUNTING-START-DATE` and `ACCOUNTING-END-DATE` fields.
- **CIMS Summary file.** The accounting dates are in the `AccountingFromDate` and `AccountingToDate` fields (starting positions 167 and 175, respectively). The `Period` and `Year` fields also reflect the accounting end dates. (These fields are at starting positions 282 and 284, respectively.) For example, if the end date is `20050916` and the date falls within the 9th period in the CIMSCalendar table, the `Period` field would contain `09` and the `Year` field `2005`.

About the Close Date

You can set an optional close date for processing input file records. Records that contain usage end dates before or after this date are processed as described in *How Accounting Dates Are Calculated* on page 3-26. The close date can be a specific date or a day of the month (the first, fifteenth, thirtieth, etc.).

You can set a system-wide close date for all process definitions and/or you can set close dates for an individual process definition or definitions. Close dates set for individual process definitions override the system-wide close date.

The close date for individual process definitions is set using the CIMSBill control statement parameter `ControlCard="DEFAULT CLOSE DAY"` in the control file (see page A-14).

The system-wide close date is set in the Configuration dialog box in CIMS Server Administrator. For more information, refer to the *CIMS Server Administrator's Guide*.

How Accounting Dates Are Calculated

The accounting dates are calculated according to the usage end dates and the close date (see *About the Close Date* on page 3-25). The accounting dates and usage end date are always the *same* in the following situations:

- If no close date is set.
- If both the CIMSBill run date and the usage end date are in the same period (as set in the CIMSCalendar table), regardless of the close date.
- If the CIMSBill run date is prior to the close date and the usage end date is in the previous period.
- If the CIMSBill run date and the usage end date are after the close date.

The accounting dates and usage end date are *different* in the following situations:

- If the CIMSBill run date is after the close date, but the usage end date is prior to the close date, the accounting dates are set to the previous day from the day CIMSBill was run.

  For example, if the first of each month is your close date and you process an input file with a usage end date of `20050930` (September 30) on October 15, the accounting start and end dates will be `20051014` (October 14).

- If the CIMSBill run date is prior to the close date and the usage end date is in a period prior to the previous period, the accounting dates are set to the last day of the previous period.

  For example, assume that the periods in the CIMSCalendar table are defined as the first day of the month to the last day of the month and that the fifteenth of each month is your close date. If you process an input file with a usage end date of `20050827` (August 27) on November 1, the accounting start and end dates will be `20051031` (October 31). October 31 is the last day of the previous period.

Account Codes and Account Code Conversion

Within chargeback and resource accounting, an account code uniquely identifies an individual, billing, or reporting entity. The actual definition of what the account code represents depends on how CIMS Server for UNIX is used. For example, an account code can represent an organization, a department in an organization, or an individual.

A key feature of CIMSAcct is its ability to use business rules to convert the identifier values in the CSR or CSR+ files into account codes. In some cases, the identifier value in the CSR or CSR+ file is the actual account code. However, in most cases, the identifier value represents the source of the consumed resources (i.e., a user ID, IP address, server name, jobname, transaction ID, terminal ID, etc.).

CIMSAcct uses an account code conversion table to convert identifier values into an account code. For example, an IP address of `10.26.12.255` is converted to the account code for Human Resources while an address of `10.26.12.260` is converted to the account code for Accounts Payable. These departments can then be charged for the consumption of the resources associated with their IP addresses.

For the steps required to set up account code conversion, see page 3-28. For an example of these steps, see page 3-36.

About Exception File Processing

During account code conversion, CIMSAcct might encounter input file records that do not match any entry in the account code conversion table. CIMSAcct includes an optional exception file processing feature that automatically identifies and removes unmatched records so that they can be reprocessed with the correct accounting information.

When this feature is enabled, all unmatched records are output to an exception file in the process definition subdirectory. You can use the exception file to identify the information that needs to be corrected, either in the records in the exception file or in the account code conversion table, and then reprocess the exception file.

To enable the exception file processing feature, see *Setting the Account Code Conversion Options* on page 3-31. For assistance in processing exception files, contact CIMS Lab.

Setting Up Account Codes and Performing Account Code Conversion

This section describes the account codes structure and how to define and convert the account code using the parameters in the control file. For an example of the account code conversion parameters in a control file, see *Control File Example* on page 3-15.

For descriptions of the control statements referenced in this section, see *Appendix A, Control Statements*.

Defining the Account Code Structure

One of the first steps in implementing CIMS Server for UNIX is deciding on a final account code structure. The structure is the chargeback hierarchy for the organization. The levels of this hierarchy are arranged from highest to lowest level within the account code. The account code can consist of up to 128 characters.

For example, your final account code structure could consist of:

DDDCCCCUUUUUUUU

D = Division (three characters) (highest level)

C = Cost Center (four characters)

U = User (eight characters) (lowest level)

Note • You can use business rules to map identifiers to applications in addition to or instead of business units, departments, cost centers, etc.

The next step is to set the account code structure in CIMS Server Administrator as described in the *CIMS Server Administrator's Guide*.

Once you have set up the account code structure, you can create an account code conversion table to convert the identifier values in the input file into an account code that fits within the final hierarchy pattern.

Defining the Account Code

The account code is derived from an identifier or identifiers in the input file. You need to specify the identifier(s) that you want to use to define the account code as follows:

- If the input file contains an `Account_Code` identifier, the `Account_Code` identifier value is automatically used to define the account code. If this value fits your account code hierarchy, you can use the value as your account code. If this value does not meet your account code requirements, you can convert the value to the appropriate account code.
- If the input file meets any of the following criteria, you need to specify an identifier or identifiers that you want to use to define the account code:
  - The input file does not contain an `Account_Code` identifier.
  - The input file contains an `Account_Code` identifier, but you want to use other identifiers instead.
  - The input file contains an `Account_Code` identifier and other identifiers that you want to use. In this case, you must define the `Account_Code` identifier in addition to the other identifiers.

  If the identifier values fit your account code hierarchy, you can use the combined values as your account code. However, in most cases you will need to convert combined values to the appropriate account code.

The identifiers that you use (either via the `Account_Code` identifier values or the identifiers that you specify) are referred to as the account code input field.

Defining the Account Code Input Field

To define the account code input field in the control file, use the `ACCOUNT FIELD` control statement parameter to define the identifiers that you want to use. Begin with `Field0` and continue sequentially as needed as shown in the following example:

```
<Paramter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
<Paramter ControlCard="ACCOUNT FIELD1,Division,1,2"/>
```

In this example, the identifiers that will be used are `UserName` and `Division`.

The numbers that follow the identifiers are the offset and length. If you want to use the full value for an identifier, use an offset of 1 and the maximum length of the identifier value as defined in the input file.

For example, assume that the maximum value for `UserName` identifier in the input file is 10 characters and that the input file contains three `UserName` values, `KSMITH`, `GBONIFANT`, and `CGREENHALL`. If you set the offset to 1 and the length to 10 and the entire identifier values would be used.

If you want to use a portion of the identifier value, use an offset position at which you want to start the value and the corresponding length. Using the `UserName` example, if you set the offset at 2 and length at 8, the identifier values used would be `SMITH`, `BONIFANT`, and `GREENHALL`.

You can define up to 10 account fields (`Field0–Field9`). The maximum number of characters for an account field is 128.

Note • Although the maximum number of characters for each account field is 128, the overall length of all account fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values (see page 3-32), cannot exceed 128 bytes.

Converting Account Code Input Field Values to Uppercase

Use the `UPPERCASE ACCOUNT FIELDS` control statement parameter to convert lowercase account code input field values to uppercase values in the resulting account code. For example, the value `ddic` would be converted to `DDIC`. By using this option, CIMSAcct account code processing becomes case-insensitive and makes defining account conversion tables much easier.

Setting the Account Code Conversion Options

To perform account conversion, regardless of whether you are using the `Account_Code` identifier or another identifier, you need to set the conversion options described in this section in the control file and create an account code conversion table as described on page 3-33. Account codes are assigned by matching identifier values in the input file records to the values in the account code conversion table.

Enabling Account Code Conversion

To enable account code conversion, use the `ACCOUNT CODE CONVERSION SORT` control statement parameter in the control file as follows:

```
ControlCard="ACCOUNT CODE CONVERSION SORT"
```

Defining the Identifier Values for Account Code Conversion

Use the `DEFINE FIELD` control statement parameter in the control file to specify the offset and length of the account code input field values that you want to use for conversion. Begin at `Field0` and continue sequentially as needed. Note that if you created an account code input field as described in *Defining the Account Code Input Field* on page 3-29, the define fields correlate to the account field values, not the original identifiers in the input file.

For example, assume that you defined the account code input field as follows:

```
<Paramter ControlCard="ACCOUNT FIELD0,UserName,2,10"/>
<Paramter ControlCard="ACCOUNT FIELD1,Division,1,2"/>
```

If you wanted to use the full values defined for the account code input field, you would use the following `DEFINE FIELD` statements:

```
<Paramter ControlCard="DEFINE FIELD0,1,10"/>
<Paramter ControlCard="DEFINE FIELD1,11,2"/>
```

The `DEFINE FIELD0` statement has an offset of 1 and length of 10 to accommodate the full 10 character length for the `UserName` identifier as defined by the `ACCOUNT FIELD0` statement.

The `DEFINE FIELD1` statement has an offset of 11 and length of 2 because the `Division` identifier starts in position 11 of the combined account code input field.

If you did not want to use the full values, you would change the offset and/or length for the define fields.

You can define up to 10 define fields (`Field0–Field9`). The maximum number of characters for an account field is 128.

Note • Although maximum number of characters for each define field is 128, the overall length of all define fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values (see page 3-32), cannot exceed 128 bytes.

Defining Optional Move Fields

Use the optional `DEFINE MOVEFIELD` control statement parameter in the control file to include either the define field values or a literal in the output account code. Begin at `Field0` and continue sequentially as needed. You can define up to 10 move fields (`Field0–Field9`). Each move field can be a maximum of 128 characters.

Note • Although the maximum number of characters for each move field is 128, the overall length of all move fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 128 bytes.

To define a define field as move field:

If you specify a define field as the move field, the define field value will be included at the beginning, end, or within the account code as defined in the account code conversion table (see *Creating an Account Code Conversion Table* on page 3-33). For example assume that you have defined define fields as follows:

```
<Paramter ControlCard="DEFINE FIELD0,1,10"/>
<Paramter ControlCard="DEFINE FIELD1,11,2"/>
```

If you want to define the entire define field `UserName` value as a move field, the control statement parameter would be as follows:

```
<Paramter ControlCard="DEFINE FIELD0,1,10"/>
<Paramter ControlCard="DEFINE FIELD1,11,2"/>
<Paramter ControlCard="DEFINE MOVEFLD0,1,10/>"
```

If you did not want to use the full values, you would change the offset and/or length for the move fields.

To define a literal as move field:

If you specify a literal as the move field, the literal value will be included at the beginning, end, or within the account code as defined in the account code conversion table (see *Creating an Account Code Conversion Table* on page 3-33). For example, if you want to define the literal `abc` as a move field, the control statement parameter would be as follows:

```
<Parameter ControlCard="DEFINE MOVEFLD0,,,abc"/>
```

Defining the Account Code Conversion Table

To define the account code conversion table that you want to use, use the parameter attribute `accCodeConvTable` in the control file. Include a path if the table is in a location other than the process definition directory in which the job script is located. For example:

`<Parameter accCodeConvTable="AcctTabl.txt"/>` (if the table is in the process definition directory)

Or

`<Parameter accCodeConvTable="/usr/local/cims/cimslab/AcctTabl.txt"/>` (if the table is not the process definition directory)

Enabling Exception Processing

Exception processing instructs the system to write all records that do not match an entry in the account code conversion table to an exception file. To enable exception file processing, include the control statement parameter `EXCEPTION FILE PROCESSING ON` in the control file as follows:

```
<Parameter ControlStatement="EXCEPTION FILE PROCESSING ON"/>
```

If this control statement is not present, the records that are not matched are written to the CIMSAcct Output CSR+ file with the unconverted account code input field value.

Considerations for Exception Processing

- If you enable exception processing, do not include a default account code as the last entry in the account code conversion table (e.g., `",,DEFAULTCODE"`). If a default account number is used, records will not be written to the exception file. For more information, see page 3-33
- The default file name for the exception file is `Exception.txt`. You should rename the output exception file so that it is not overwritten when subsequent files are written.

Creating an Account Code Conversion Table

The account code conversion table is an ASCII text file that contains the definitions required to convert the identifier values defined by the account code input field to user-defined output account codes as shown in the following example:

```
ABC,DEF,ATM@0
GHI,JKL,COM@0
MNO,PQR,MTG@0
```

The fields in the account code conversion table entries are low identifier, high identifier, the account code that you want to convert to (maximum of 128 characters), and an optional move field indicator, which can be placed before, after, or within the account code. In the preceding example, the move field value will appear at the end of the account code. For an example of how to create a account code conversion table, including the use of a move field, see *Account Code Conversion Example* on page 3-36.

Consider the following when creating the account code conversion table:

- The account code conversion table is case-sensitive. For convenience, you can enter uppercase values in the table and use the `UPPERCASE ACCOUNT FIELDS` control statement parameter. This is especially helpful it you are using one account code conversion table for multiple process definitions. This ensures that account code input field values that are lowercase or mixed case are processed.
- Each record in the table can be up to 386 characters (129 for the low identifier, 129 for the high identifier, and 128 for the new account code).
- The identifier fields follow these rules:
  - Each low and high identifier field is compared to the corresponding account code input field values. If the compares are true, the account code is assigned.
  - The low identifier fields are padded with x'00' and the high value fields are padded with x'FF'.
  - The high identifier field is set equal to the low field plus the high padding when the high identifier value is null.
  - You can enter up to 10 levels of 128 characters each for the low and high identifier fields. Each level must be delimited by a colon (:).

  Note • Although the maximum number of characters for each field level is 128, the overall length of all levels added together cannot exceed 128 characters.

  For example, if you type AA as the low identifier and CC as the high identifier, all account codes that begin with characters greater than or equal to AA or less than or equal to CC are converted to the specified account code. If you type AAAAAAAA:BB as the low identifier and CCCCCCCC:DD as the high identifier, all account codes that begin with characters greater than or equal to AAAAAAAABB or less than or equal to CCCCCCCCDD are converted to the specified account code.
- You can include a default account code as the last entry in the account code conversion table by leaving the low and high identifier fields empty (e.g., "`,,DEFAULTCODE`"). In this case, all records that contain identifier values that do not match an entry in the account code conversion table will be matched to the default code.

- If you do not include a default account code, input file records that contain identifier values that do not match an entry in the account code conversion table are written to the CIMSAcct Output CSR+ file with the unconverted account code input field value as the account code.

  If you want unmatched records to go to an exception file for reprocessing at a later time, you need to use the `EXCEPTION FILE PROCESSING ON` control statement parameter. If you enable exception file processing, make sure that you have not included a default account code entry in the account code conversion table.

- If the `PRINT ACCOUNT NO-MATCH` statement is included in the control file, a message is written to STDOUT or the CIMSAcct Message file showing the identifier values from the input file that were not matched during account code conversion. If this statement is not present, only the total number of unmatched records is provided. The system prints up to 1000 messages.

  Note • The `messageFile` parameter attribute in the control file determines whether messages are written to STDOUT or the CIMSAcct Message file. See page 3-18.

- The number of definition entries that you can enter in the table is limited only by the memory available to CIMSAcct.

Account Code Conversion Example

Note • This section provides examples to supplement the preceding sections. Please refer to these sections for more detailed information.

Assume that you want to use the `User` identifier values in an input file to define your account code and that your account code is in the following structure:

DDDCCCCUUUUUUUU

D = Division (three characters) (highest level)

C = Cost Center (four characters)

U = User (eight characters) (lowest level)

The following table shows the `User` identifier values that are in the input file records and the corresponding account code that you want to convert the value to.

User Identifier Value from Input File		Final Account Code		Division Account Code	Cost Center Account Code	User Account Code
DDIC	=	FINACTGDDIC	=	FIN	ACTG	DDIC
SAP	=	ISDDEVPSAP	=	ISD	DEVP	SAP
SAPCOMM	=	OPSMKTGSAPCOMM	=	OPS	MKTG	SAPCOMM
SAPSYS	=	MISPRODSAPSYS	=	MIS	PROD	SAPSYS
TEAM718	=	MISTESTTEAM718	=	MIS	TEST	TEAM718
TEAMF99	=	OPSTESTTEAMF99	=	OPS	TEST	TEAMF99

To convert the `User` identifier values to the final account code, you would complete the steps in the following sections.

Define the Account Code and Conversion Parameters in the Control File

In the example on page 3-36, the longest value for the User identifier is 7 characters (SAPCOMM, TEAM718, and TEAMF99). To use the entire value, you need to define the account input field and the define field with an offset of 1 and a length of 7 as follows:

```
<Parameter ControlCard="Account Code Conversion Sort"/>
<Parameter ControlCard="ACCOUNT FIELD0, User, 1, 7"/>
<Parameter ControlCard="DEFINE FIELD0, 1, 7"/>
<Parameter ControlCard="DEFINE MOVEFLD0, 1, 7"/>
<Parameter ControlCard="EXCEPTION FILE PROCESSING ON"/>
```

Because you also want to include the define field value at the end of the account code, you need to define a move field for the value, also with an offset of 1 and a length of 7.

Note that the ACCOUNT CODE CONVERSION SORT control statement parameter is required to enable account code conversion. The EXCEPTION FILE PROCESSING ON control statement parameter instructs CIMSAcct to produce an exception file.

Create the Account Code Conversion Table

The account code conversion table would contain the following entries. Note that the high identification code is not needed in this example because the low and high identification codes are the same.

```
DDIC,,FINACTG@0
SAP,,ISDDEVP@0
SAPCOM,,OPSMKTG@0
SAPSYS,,MISPROD@0
TEAM718,,MISTEST@0
TEAMF99,,OPSTEST@0
```

The `@0` at the end of the account code value specifies that the move field defined in the previous step will be included at the end of the account code. For example, the resulting account code for the first table entry would be `FINACTGDDIC`. If you placed the `@0` at the beginning of the account code value, the resulting account code would be `DDIC<three spaces>FINACTG`. You could place the `@0` anywhere in the account code.

If you had additional move fields defined, they would be represented by `@1`, `@2`, `@3`, etc. For example, if you had used another identifier for account conversion in addition to `User` and had defined move fields for each (move field 0 and move field 1), `@1FINACTG@0` would place the value for move field 1 at the beginning of the account code and the value for move field 0 at the end of the account code.

Setting Up Shifts

In CIMS Server Administrator, you can set rates for a rate code by shift. This optional feature enables you to set different rates based on the time of day. For more information about setting rates by shift, refer to the *CIMS Server Administrator's Guide*.

You can use either of the following methods to determine the shift codes that are used for processing. These shift codes appear in the CIMSAcct Output CSR+ records.

- Use the shift code from the input file records. If a shift code is not included in the records, the default shift code is 1.
- If you do not want to use the shift codes in the input record for processing, you can recalculate the shifts using the start date/time in the records and the CIMSAcct control statement parameter `SHIFT` as described in *Using the SHIFT Control Statement* on page 3-39.

Important! • Regardless of whether you use the rate codes from the input file or recalculate the rate codes, you need to include the CIMSBill control statement parameter `USE SHIFT CODES` in the control file. If this control statement is not present, CIMSBill uses the default rate shift 1 regardless of the shift code that appears in the CIMSAcct Ouput CSR+ file.

Using the SHIFT Control Statement

You can use the CIMSAcct control statement parameter `SHIFT` in the control file to enter daily shifts. You can include a maximum of seven `SHIFT` control statement parameters (one for each day of the week). Up to 9 shifts can be specified for each `SHIFT` statement.

CIMSAcct compares the start dates and times in the input records to the daily shifts that you specify using the `SHIFT` control statement parameters and calculates new shift codes. The recalculated shift codes rather than the original shift codes appear in the CIMSAcct Ouput CSR+ file.

The syntax for the SHIFT control statement is as follows

```
SHIFT [day] [code] [end time] [code] [end time] …. [code] [end time]
```

`day` = SUN | MON | TUE | WED | THU | FRI | SAT

`code` = 1–9 (the shift code)

`end time` = the end time of the shift

The shift end times must be listed in 4-character, 24-hour format.

Note • CIMSAcct will not recalculate the shift codes for CSR or CSR+ files produced by CIMS Mainframe Data Collector and Chargeback System. CIMS Mainframe records are aggregated using the shift code as one of the aggregation points. Therefore, the time in the aggregated record is not representative of every record within the aggregation. For more information, refer to the *CIMS Mainframe Data Collector and Chargeback System User Guide*.

Control File Example With Shifts

In this example, three shifts have been defined for each day of the week. Shift 3 is from midnight to 8 a.m., shift 1 is from 8 a.m. to 4 p.m., and shift 2 is from 4 p.m. to midnight. Note that the required CIMSBill control statement parameter `USE SHIFT CODES` is included.

```
<?xml version="1.0" encoding="utf-8" ?>
<CIMSProcessing version= "4.0">
   <Step id="Process Acct"
      programName="CIMSAcct">
      <Parameters>
        <Parameter accCodeConvTable="AcctTabl.txt"/>
        <Parameter FullMessageFormat="True"/>
        <Parameter MessageFile="AcctMsg.log"/>
        <Parameter trace="false"/>
        <Parameter ControlCard="Account Code Conversion Sort"/>
        <Parameter ControlCard="Print Account No-Match"/>
        <Parameter ControlCard="ACCOUNT FIELD0,SYSTEM_ID,1,8"/>
        <Parameter ControlCard="ACCOUNT FIELD1,OR_BASE,1,8"/>
        <Parameter ControlCard="DEFINE FIELD0,1,8"/>
        <Parameter ControlCard="DEFINE FIELD1,9,8"/>
        <Parameter ControlCard="DEFINE MOVEFLD0,1,8"/>
        <Parameter ControlCard="DEFINE MOVEFLD1,9,8"/>
        <Parameter ControlCard="SHIFT SUN 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT MON 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT TUE 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT WED 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT THU 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT FRI 3 0800 1 1600 2 2400"/>
        <Parameter ControlCard="SHIFT SAT 3 0800 1 1600 2 2400"/>
     </Parameters>
    </Step>
   <Step id="Process Bill"
      programName="CIMSBill">
      <Parameters>
        <Parameter Full MessageFormat="true"/>
        <Parameter MessageFile="BillMsg.log"/>
        <Parameter trace="false"/>
        <Parameter ControlCard="USE SHIFT CODES"/>
      </Parameters>
   </Step>
   <Step id="Load Detail"
     programName="CIMSLoad">
       <Parameters>
         <Parameter LoadType="ALL"/>
         <Parameter BulkLoad="True"/>
         <Parameter detailFile="BillDetail.txt"/>
         <Parameter identFile="Ident.txt"/>
         <Parameter summaryFile="BillSummary.txt"/>
         <Parameter trace="true"/>
       </Parameters>
    </Step>
</CIMSProcessing >
```

Normalizing CPU Values

Computers within an organization have different processing speeds. This speed difference might cause users to request that their work be run on the faster machine to reduce costs. This situation could lead to heavy workloads on the faster computers while the slower units stand idle. To avoid this problem, you can normalize the processing speeds to more evenly charge for CPU utilization. That is, you can assign a percentage of the original CPU time used by a system rather than the actual time to be used during the billing process.

For example, your organization has two z/OS systems, AL90 and AL95. System AL95 is 20 percent faster than AL90. If you use AL95 as the base system, use a factor of .80 to normalize AL90 to reflect the speed of AL95.

Note • Due to the disparity between the way different operating systems capture performance statistics, it is not desirable to normalize the processor times between platforms (e.g., z/OS to UNIX or UNIX to Windows).

To perform normalization, you need to do the following:

- In the control file that contains the CIMSBill parameters (see page 3-15), include the `NORMALIZE CPU VALUES` control statement parameter.
- In CIMS Server Administrator, do the following:
  - Select the CPU Value option for the rate code or codes for which you want to perform normalization.
  - Set the normalization factor for the applicable system or systems.

To set the rate code as a CPU value:

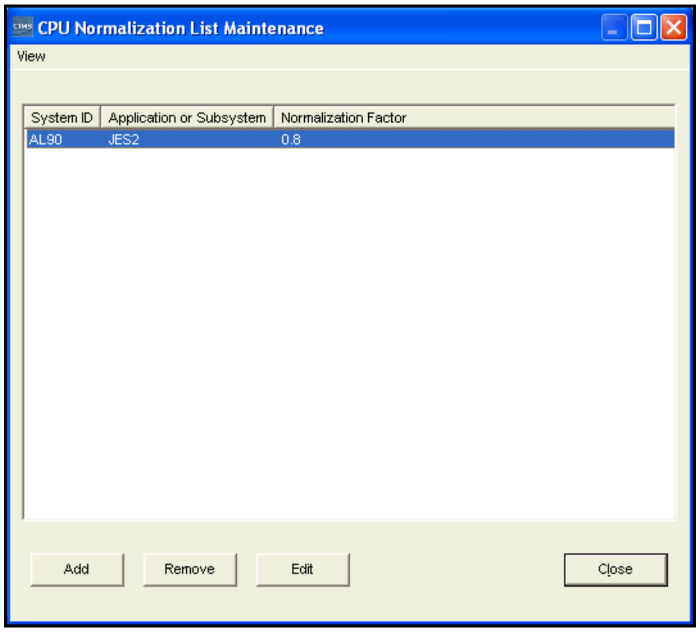
1. In the CIMS Server Administrator main window, click **Chargeback Administration** ▸ **Chargeback Table Maintenance** ▸ **Rate Codes**.
2. In the Rate Code Maintenance List dialog box, click the rate code that you want to set as a CPU value, (for example, Mainframe CPU Minutes [Z003]), and then click **Edit.**
3. Select **CPU Value** check box, and then click **OK**.

   Note • If this item is not checked, the rate code value will not be normalized during CIMSBill processing.

To set the normalization factor by system:

1. In the CIMS Server Administrator main window, click **Chargeback Administration ▸ Processing ▸ CPU Normalization**.
2. Click **Add** in the CPU Normalization Maintenance List dialog box.
3. Enter the following:
   - **System ID**. The name of the system. For z/OS, this is the four-character System Model ID, for UNIX and Windows, it is the computer name. Note that the input file must contain an identifier named `System_ID` to identify the system name for CIMSBill.
   - **Work ID**. The subsystem name. This is any other system value that can narrow the normalization (i.e., the CICS region name, the DB2 plan name, the Oracle instance, etc.). Note that the input file must contain an identifier named `Work_ID` to identify the subsystem name for CIMSBill.
   - **Factor**. This is the actual normalization factor. In the example at the beginning of this section, this would be .80.
4. Click **OK**.

An example of a configured CPU Normalization List Maintenance dialog box is shown in Figure 3-1.



System ID	Application or Subsystem	Normalization Factor
AL90	JES2	0.8

Figure 3-1 • CPU Normalization List Maintenance Dialog Box

To edit or remove a factor, click the factor, and then click **Edit** or **Remove**. Note that you cannot change the System or Work ID when you edit the factor.

Include/Exclude Processing

By default, CIMSBill processes all records within the input CIMSAcct Output CSR+ file. However, you might want to process only certain records or you might want to exclude certain records. This is useful if there is data that your organization does not want to report on.

You can exclude records by date using the `DATE SELECTION` control statement parameter for CIMSAcct or CIMSBill.

You can also include or exclude records by identifier values in the records. For example, you might want to exclude all records that contain a particular `Jobname` identifier value.

To include or exclude records by identifier values, you need to include the CIMSBill control statement parameters `INCLUDE` or `EXCLUDE` in the control file.

Example

Assume that you have a CIMSAcct Output CSR+ from a DB2 system with the following identifiers:

```
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI09999,Jobname,BKMALI01,Start_date,20050714...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI09999,Jobname,BKMALI02,Start_date,20050715...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI09999,Jobname,BKMALI03,Start_date,20050716...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALE09999,Jobname,DFMALE01,Start_date,20050714...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALE09999,Jobname,DFMALE02,Start_date,20050715...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALE09999,Jobname,DFMALE03,Start_date,20050716...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALU09999,Jobname,LCHALU01,Start_date,20050714...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALU09999,Jobname,LCHALU02,Start_date,20050715...
...System_ID,ALIJ,Work_ID,JES2,Account_Code,ALU09999,Jobname,LCHALU03,Start_date,20050716...
```

You want to exclude those records that contain `DFM` in the `Jobname` identifier value, you need to include the following control statement parameter in the control file:

```
<Parameter ControlCard="EXCLUDE,S390DB2,JOBNAME,1,3,DFM,DFM"/>
```

The syntax for this control statement is as follows:

- Record header (in this example, `S390DB2`)

  Note • For CSR+ files, use the second record header in the file. The first header is used for sorting.

- Identifier value (in this example, `JOBNAME`)
- Offset (in this example, `1`)
- Length (in this example, `3`)
- Low identifier value (in this example, `DFM`)
- High identifier value (in this example, `DFM`)

Note • The high identifier is required even if it is the same as the low identifier.

Running CIMS Server for UNIX

This section describes how to run CIMS Server for UNIX. You should determine the frequency that you want to process data as described in *Running CIMS Server for UNIX Using the nightly.sh Script* before you run CIMS Server for UNIX.

Running CIMS Server for UNIX Using the nightly.sh Script

You can run CIMS Server for UNIX using the `nightly.sh` script described on page 2-11.

The `nightly.sh` script must be scheduled to run on a regular basis. You can use any batch scheduler to run this script. CIMS Lab recommends that you run the script using the CIMS user account designated during the installation of CIMS Server for UNIX.

Example

To schedule the `nightly.sh` script to run every morning at 4:15, the following crontab entry could be used:

```
15 4 * * * /usr/cims/scripts/nightly.sh
```

If an error occurs and the CIMS Server for UNIX job(s) are not run (for example, the job script contains a syntax error) a job log file is not created and e-mail notification of the job failure is not sent. To ensure that jobs run correctly, you might want to run the CIMS Server for UNIX from the command line before using the `nightly.sh` script to run the program.

Running CIMS Server for UNIX From the Command Prompt

At the `/usr/local/cims/cimlal/jlib>` prompt, type `java -jar CIMSAcct.jar` followed by the process definition directory and control file parameters. Or from any prompt, type `java -jar /usr/local/cims/cimlab/jlib/CIMSAcct.jar` followed by the parameters.

This same syntax is used for the CIMSBill and CIMSLoad programs.

Examples

```
/usr/local/cims/cimlab/jlib> java -jar CIMSAcct.jar /usr/local/cims/cimlab/
processes/UnixDB2 ProcCntl.txt
```

Or

```
java -jar /usr/local/cims/cimlab/jlib/CIMSAcct.jar /usr/local/cims/cimlab/
processes/UnixDB2 ProcCntl.txt
```

In these examples, the process definition directory is `/usr/local/cims/cimlab/processes/UnixDB2` and the control file is `ProcCntl.txt`.

4

Database Administration

About Loading and Maintaining the Database

CIMS Server for UNIX provides a flexible, cross-platform architecture that supports multiple database types. The components used to load and maintain the database are the CIMS Processing Engine and CIMS Server Administrator applications, respectively.

Using CIMS Processing Engine

CIMS Processing Engine is used to process input CSR or CSR+ files and load output data into the database. CIMS Processing Engine runs on either the Windows or UNIX platform.

To run the engine on the Windows platform, you need to use the COM-based engine described in the *CIMS Server Administrator's Guide*.

To run the engine on the UNIX platform, you need to use the Java-based engine included in the CIMS Server for UNIX installation.

Depending on the version of CIMS Processing Engine that you are using (Windows or UNIX), you need to set the processing options as follows:

- If you are using CIMS Processing Engine for the Windows platform, use the Windows-based CIMS Server Administrator application to set up and maintain all settings related to process definitions and processing data. For more information, refer to the *CIMS Server Administrator's Guide*.

- If you are using CIMS Processing Engine for the UNIX platform, the files and scripts described in this guide are used to set up, maintain, and run the engine. At this time, there is no GUI interface for the UNIX version of the engine. The process definition path in the CIMS Server Administrator Configuration dialog box and tasks described in the *Processing Data* chapter and other chapters of the *CIMS Server Administrator's Guide* are not applicable for the UNIX version of the engine.

Using CIMS Server Administrator

The CIMS Server Administrator application is used to set up and maintain the data in the database tables (i.e., the CIMSClient table, CIMSRate table, CIMSUser table, etc.) regardless of the platform that the database resides on or the version of CIMS Processing Engine that you are using (UNIX or Windows). Refer to the *CIMS Server Administrator's Guide* to set up and maintain the information in database tables.

Creating a Database

You can use more than one database with CIMS Server for UNIX. For example, you might have a production database and a development database.

To create an Oracle database or databases, consult your Oracle DBA. For more information about the Oracle database requirements, see *Enabling CIMS Server to Access an Oracle Database* on page 4-4.

To create a SQL Server database or databases, follow the steps in *CIMS Server Administrator's Guide*.

Important! • Although the *CIMS Server Administrator's Guide* provides instructions for creating a Windows authenticated user login for SQL Server, you cannot use Windows authentication if you are using the UNIX version of CIMS Processing Engine. You must use a SQL Server authenticated user login.

To create a DB2 database, consult your DB2 DBA. For more information about the DB2 database requirements, see the *CIMS Server for DB2 Database Setup Guide*.

Administering the Database

The administrative tasks required to set up and maintain the database are performed using the CIMS Server Administrator application. For information specific to setting up and administering the data in the database, refer to the *CIMS Server Administrator's Guide*.

Enabling CIMS Server to Access an Oracle Database

To enable CIMS Server to access an Oracle database, you need to complete the steps in the following sections.

Install the Oracle Client

The Oracle Client must be installed on the client computer(s). The client installation includes OO4O (Oracle Objects for OLE), which is required to access the Oracle database.

Configure Connection Between Servers

Configure a connection between the CIMS Server application server and the Oracle database server.

Enable Read Permission for the ASP.NET Local User

On the CIMS Server application server, make sure that the ASP.NET local user has a minimum security permission of Read for the Oracle home folder and that the permission is propagated down through all subfolders and files as follows:

1. Right-click the Oracle home folder, and then click **Sharing and Security**.

   The **Properties** dialog box appears.

2. On the **Security** tab, click **Add**.

   The Select Users, Computer, or Group dialog box appears.

3. Click **Locations**, click the local computer name, and then click **OK**.

4. In the **Enter object name to select** box, type `aspnet`, and then click **OK**.

   The user ASP.NET Machine Account (*computer name*\ASPNET) appears in the **Group or user names** box with permissions of Read & Execute, List Folder Contents, and Read.

5. Click the ASP.NET user, and then click **Advanced**.

   The Advanced Security Options dialog box appears.

6. On the **Permissions** tab, make sure that the ASP.NET user is selected, and then select both the **Inherit from parent...** and **Replace permission entries on all child objects...** check boxes, and then click **Apply**.

7. Click **Yes** in the Security message box.

8. When the security settings update completes, click **OK** to close the Advanced Security Settings dialog box.

9. Click **OK** to close the Properties dialog box.

Modify the sqlnet.ora File

The IIS Web server used for Web reporting can be set up to use either anonymous or Windows authenticated access. (For information regarding setting up and configuring the Web server, refer to the *CIMS Server Administrator's Guide*.)

If you are using an Oracle database and want to use Windows authentication for IIS, you need to modify the following line in the `...\oracle\ora92\network\ADMIN\sqlnet.ora` file:

```
SQLNET.AUTHENTICATION_SERVICES= (NTS)
```

Either comment out the line by placing a pound sign (#) at the beginning of the line or change `NTS` to `NONE` as shown in the following examples:

```
#SQLNET.AUTHENTICATION_SERVICES= (NTS)
```

Or

```
SQLNET.AUTHENTICATION_SERVICES= (NONE)
```

If this line is not modified, the following error is issued: `ORA-12638: Credential retrieval failed`.

Create a CIMS Data Source for the Database

You need to create a CIMS Data Source that enables the CIMS Server to access the database. This is a different procedure than creating a CIMS Data Source for the UNIX version of CIMS Processing Engine as described in *Modifying the CIMSDBConfig.xml File* on page 1-14.

To set up the CIMS Data Source, start CIMS Server Administrator and follow the data source configuration steps specific to the Oracle database in Chapter 1 of the *CIMS Server Administrator's Guide*.

Create a Schema Log On Trigger for the CIMS User

When you create a CIMS Data Source in CIMS Server Administrator, you are asked to associate a user with the schema for the CIMS database. However, the user ID and schema defined for the CIMS Data Source are not applied to all CIMS administrative options, CIMS scripts, or to Crystal Reports. To ensure that this user ID and schema are used for all CIMS components, you need to create a trigger that changes the user's default schema to this schema.

To create and install the trigger, you can use the `OracleLogonTrigger.bat` script as described in Appendix C of the *CIMS Server Administrator's Guide*.

Note • If the user ID that you are using to access the CIMS database is the same as the schema for the database, this step is not required.

A

Control Statements

This appendix describes the CIMSAcct and CIMSBill control statements. Control statements are defined as parameters in the control file (see *Setting Up the Control File (ProcCntl.xml)* on page 3-15).

CIMSAcct Control Statements

This section describes the CIMSAcct control statements.

Note • Control statements are not case-sensitive, except for the literals in the `DEFINE MOVEFLD` **statement (see page A-8).**

CONTROL STATEMENT	SEE	DESCRIPTION
`ACCOUNT CODE CONVERSION {SORT}`	[page A-3]	Specifies processing of the CIMS account code conversion table.
`ACCOUNT FIELD`	[page A-4]	Defines how to build the account code from the identifiers within the input file record.
`DATE SELECTION`	[page A-5]	Defines a date range for records to be processed by CIMSAcct.
`DEFINE FIELD`	[page A-6]	Used only during account code conversion. Defines the fields for comparison within the account code input field (which is built from the `ACCOUNT FIELD` statements or from the identifier `Account_Code`).
`DEFINE MOVEFLD`	[page A-7]	Used only during account code conversion. Defines the fields that are to be moved from the account code input field (which is built from the `ACCOUNT FIELD` statements or from the identifier `Account_Code`) to the account code output field.
`EXCEPTION FILE PROCESSING ON`	[page A-8]	Used only during account code conversion. When present, any record that does not match an entry in the account code conversion table is written out to the exception file.
`PRINT ACCOUNT NO-MATCH`	[page A-8]	Prints the first 1000 input identifiers that are not matched during account code conversion.
`SHIFT`	[page A-9]	Sets up the shifts within the system.
`UPPERCASE ACCOUNT FIELDS`	[page A-10]	Specifies that the account code built from the account code input field is uppercase.

ACCOUNT CODE CONVERSION {SORT}

Format:

```
ACCOUNT CODE CONVERSION {SORT}
```

This statement enables account code conversion using and `DEFINE FIELD` and `DEFINE MOVEFLD` statements and the account code conversion table. The `SORT` option is recommended because it sorts the account code conversion table in memory and increases performance.

When account code conversion is enabled, all no-match records are written to the CIMSAcct Output CSR+ file with the unconverted account code input field value as the account code. If you want the record written to an exception file for later processing, you must add the control statement `Exception File Processing On`.

Example:

```
ACCOUNT CODE CONVERSION SORT
```

ACCOUNT FIELD

Format:

```
ACCOUNT FIELDn,identifier_name,offset,length
```

`n` = 0–9 (up to 10 `ACCOUNT FIELD` statements supported)

`offset` = starting offset into identifier value (1–255)

`length` = identifier value length from offset (1–128)

Note • Although the maximum number of characters for each account field is 128, the overall length of all account fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values (see page A-7), cannot exceed 128 bytes.

This statement defines the identifier(s) that you want to use to build the account code. Begin at `ACCOUNT FIELD0` and continue sequentially as needed. The account fields are used (along with `DEFINE FIELD` and `DEFINE MOVEFLD` statements) in account code conversion if account code conversion is enabled. If account code conversion is not enabled, then the account code is built directly from this statement or from the `Account_Code` identifier in the input file records (if present). For more information, see see *Defining the Account Code* on page 3-29.

Example:

```
ACCOUNT FIELD0,UserName,1,10
ACCOUNT FIELD1,Division,1,2
```

In this example, the `UserName` and `Division` identifiers will be used to create the account code. The offset for both identifier values is `1` and the length of the values are `10` and `2`, respectively. Therefore, the account code will be the first 10 characters of the user name followed by the first 2 characters of the division.

If the `UPPERCASE ACCOUNT FIELDS` statement is specified, lowercase identifier values in the account field are converted to uppercase values.

DATE SELECTION

Format:

```
DATE SELECTION {YYYYMMDD YYYYMMDD | keyword}
```

This statement defines a date range for records to be processed by CIMSAcct. Records are selected by the usage end date in the record. You can use the following values:

- From and to dates. For a record to be selected, it must be greater than or equal to the from date and less than or equal to the to date.

  or

- One of the following keywords:

Keyword	Description
`**RNDATE`	Selects records based on the run date
`**CURDAY`	Selects records based on the run date and the run date less one day
`**CURWEK`	Selects records based on the run week (Sun–Sat)
`**CURMON`	Selects records based on the run month
`**PREDAY`	Selects records based on the run date, less one (day)
`**PREWEK`	Selects records based on the previous week (Sun–Sat)
`**PREMON`	Selects records based on the previous month
`CURRENT`	Selects current period from the CIMSCalendar table
`PREVIOUS`	Selects previous period from the CIMSCalendar table **Note:** CIMSAcct requires access to the database if the `CURRENT` or `PREVIOUS` keyword is present.

The start and end dates of data processed through CIMS Server must reflect the start and end dates of the target billing period. If the billing period runs from January 1 through January 31, any records produced with a date of February 1 (or later) should not be included as part of the processing. In most cases, this data screening has already been performed, either by CIMS Server or through the creation of data inputs reflecting the desired date range.

Note that it is rare that you would need to limit the date range of the data used as input to CIMSAcct. `DATE SELECTION` is usually used as CIMSBill control statement.

Examples:

```
DATE SELECTION 20050601 20050630
```

```
DATE SELECTION **PREMON
```

DEFINE FIELD

Format:

```
DEFINE FIELDn,offset,length
```

`n` = 0–9 (up to 10 `DEFINE FIELD` statements supported)

`offset` = starting offset into account field or the `Account_Code` identifier value (1–128)

`length` = account field or identifier value length from offset (1–128)

This statement is used only during account code conversion. Use this statement to specify the offset and length of the account code input field value that you want to use for conversion. Begin at `DEFINE FIELD0` and continue sequentially as needed.

Note • Although the maximum number of characters for each define field is 128, the overall length of all account fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values (see page A-7), cannot exceed 128 bytes.

For more information about using define fields, see *Defining the Identifier Values for Account Code Conversion* on page 3-31.

Example:

```
ACCOUNT FIELD0,UserName,1,10
ACCOUNT FIELD1,Division,1,2
DEFINE FIELD0,3,8
DEFINE FIELD1,11,2
```

In this example, the first two characters of `ACCOUNT FIELD0` *will not* be considered for account code conversion as specified by the offset of `3` for `DEFINE FIELD0`. However, the full value of `ACCOUNT FIELD1` will be used because the offset for `DEFINE FIELD1` is 11 (the `Division` identifier value starts in position 11 of the combined account fields) and the length for both the account field and define field are the same.

For example, if the combined value for `ACCOUNT FIELD0` and `ACCOUNT FIELD1` is `AABBCCDDEEXZ`, the value as defined by the `DEFINE FIELD0` and `DEFINE FIELD1` is `BBCCDDEEXZ`.

DEFINE MOVEFLD

Format:

```
DEFINE MOVEFLDn,offset,length,literal
```

`n` = 0–9 (up to 10 `DEFINE MOVEFLD` statements supported)

`offset` = starting offset into account field or the `Account_Code` identifier value (1–128)

`length` = account field or identifier value length from offset (1–128)

`literal` = a literal can be moved instead of a particular field (literals in this statement are case-sensitive)

This statement is used only during account code conversion. Use this statement to move all or a portion of the account code input field value to a position in the output account code. Begin at `DEFINE MOVEFLD0` and continue sequentially as needed.

Targets within the account code conversion table are specified as `@0–@9`.

Note • Although the maximum number of characters for each move field is 128, the overall length of all account fields added together cannot exceed 128 characters.

Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 128 bytes.

For more information about using move fields, see *Defining Optional Move Fields* on page 3-32.

Example:

```
ACCOUNT FIELD0,UserName,1,10
ACCOUNT FIELD1,Division,1,2
DEFINE FIELD0,3,8
DEFINE FIELD1,11,2
DEFINE MOVEFLD0,11,2
```

In this example, the value for `DEFINE MOVEFLD0` specifies that the value beginning at position 11 account code input field will be placed in the output account code as defined by the account code conversion table.

For example, if the 2-character value beginning at position 11 is `XZ` and the account code conversion table is:

BBC,,FINACCT@0

The resulting account code will be `FINACCTXZ`.

For more information about the account code conversion table, see *Creating an Account Code Conversion Table* on page 3-33.

EXCEPTION FILE PROCESSING ON

Format:

```
EXCEPTION FILE PROCESSING ON
```

This control statement is used only during account code conversion. When this statement is present, any record that does not match an entry in the account code conversion table is written to an exception file. If this control statement is not present, the records that are not matched are written to the CIMSAcct Output CSR+ file with the unconverted account code input field value as the account code.

For more information the exception file processing, see *Enabling Exception Processing* on page 3-33.

Example:

```
EXCEPTION FILE PROCESSING ON
```

Considerations

- If you enable exception processing, do not include a default account code as the last entry in the account code conversion table (e.g., `",,DEFAULTCODE"`). If a default account number is used, records will not be written to the exception file.
- The default file name for the exception file is `Exception.txt`. You should rename the output exception file so that it is not overwritten when subsequent files are created.

PRINT ACCOUNT NO-MATCH

Format:

```
PRINT ACCOUNT NO-MATCH
```

When this statement is present, a message is written to STDOUT or the CIMSAcct Message file showing the identifier values from the input file that were not matched during account code conversion. If this statement is not present, only the total number of unmatched records is provided. The system prints up to 1000 messages.

Note • The `messageFile` parameter attribute in the control file determines whether messages are written to STDOUT or the CIMSAcct Message file. See page 3-18.

SHIFT

Format:

`SHIFT [day] [code] [end time] [code] [end time] …. [code] [end time]`

`day` = SUN | MON | TUE | WED | THU | FRI | SAT

`code` = 1–9 (the shift code)

`end time` = the end time of the shift

This control statement defines shifts within the system. Seven shift records are supported (one for each day of the week) and up to nine shifts can be specified on each. End times are entered in hours and minutes using the 24-hour clock.

Rules that apply to shift records:

- Day is defined as the first three letters of the day.
- Up to nine shifts per day can be specified.
- The preceding end time must always be less than the next end time.
- No shift spans midnight.
- If you define shifts for one or more days of the week, but not all days, the shifts for the remaining days are defined as follows by default:

  ```
  3 0800 1 1630 2 2400
  ```

Example:

For the following shifts:

Monday through Friday

Shift 1 - 5 AM to 8 AM and 3:30 PM to 5 PM

Shift 2 - 8 AM to 11:30 AM and 1:30 PM to 3:30 PM

Shift 3 - 5 PM to 8 PM

Shift 4 - 9:30 PM to 12 AM and 12 AM to 5 AM

Shift 5 - 11:30 AM to 1:30 PM and 8 PM to 9:30 PM

Saturday and Sunday

Shift 1 - 8 AM to 5 PM

Shift 2 - 5 PM to 12 AM and 12AM to 8 AM

The `SHIFT` statements are as follows:

```
SHIFT SUN 2 0800 1 1700 2 2400

SHIFT MON 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400

SHIFT TUE 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400

SHIFT WED 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400

SHIFT THU 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400

SHIFT FRI 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400

SHIFT SAT 2 0800 1 1700 2 2400
```

UPPERCASE ACCOUNT FIELDS

Format:

```
UPPERCASE ACCOUNT FIELDS
```

This control statement instructs CIMSAcct to convert lowercase account code input field values to uppercase values in the resulting account code. For example, the value `ddic` would be converted to `DDIC`. By using this option, CIMSAcct account code processing becomes case-insensitive and makes defining account conversion tables much easier.

CIMSBill Control Statements

This section describes the CIMSBill control statements.

CONTROL STATEMENT	SEE	DESCRIPTION
`BACKLOAD DATA`	[page A-12]	Instructs CIMSBill to ignore the close date. The accounting dates created by CIMSBill are the same as the usage end date regardless of the close date
`CLIENT SEARCH ON`	[page A-12]	Instructs CIMSBill to use the account code structure definition to search the CIMSClient table for the rate code table associated with a client.
`DATE SELECTION`	[page A-13]	Defines a date range for records to be processed by CIMSBill.
`DEFAULT CLOSE DAY`	[page A-14]	Overrides the system-wide close date.
`DEFINE`	[page A-14]	Defines the account code structure used by the statements `CLIENT SEARCH ON` and `DYNAMIC CLIENT ADD ON`.
`DYNAMIC CLIENT ADD ON`	[page A-15]	Specifies that CIMSBill automatically insert corresponding client entries into the CIMSClient table for all clients that have a resource usage charge.
`EXCLUDE`	[page A-15]	Specifies an exclude record condition.
`INCLUDE`	[page A-16]	Specifies an include record condition.
`KEEP ORIGINAL CPU VALUES`	[page A-16]	Specifies whether to include the original CPU value as an identifier.
`NORMALIZE CPU VALUES`	[page A-16]	Normalizes CPU values.
`REPORT DATE`	[page A-17]	Specifies the accounting dates that are put into the Summary and Detail records created by CIMSBill.
`USE SHIFT CODES`	[page A-18]	Instructs CIMSBill to use the shift character specified in the CIMSAcct Output CSR+ record and apply the appropriate rate shift value.

BACKLOAD DATA

Format:

```
BACKLOAD DATA
```

This statement instructs CIMSBill to ignore the close date. The accounting dates created by CIMSBill are the same as the usage end date regardless of the close date.

CLIENT SEARCH ON

Format:

```
CLIENT SEARCH ON
```

This statement instructs CIMSBill to use the account code structure definition (by use of the `DEFINE Jn` fields, see page A-14) to search the CIMSClient table for the rate code table associated with a client. If this statement is not specified, the STANDARD rate code table is used for all clients.

DATE SELECTION

Format:

```
DATE SELECTION {YYYYMMDD YYYYMMDD | keyword}
```

This statement defines a date range for records to be processed by CIMSBill. Records are selected by the accounting end date in the record. You can use the following values:

- From and to dates. For a record to be selected, it must be greater than or equal to the from date and less than or equal to the to date.

  or

- One of the following keywords:

Keyword	Description
`**RNDATE`	Selects records based on the run date
`**CURDAY`	Selects records based on the run date and the run date less one day
`**CURWEK`	Selects records based on the run week (Sun–Sat)
`**CURMON`	Selects records based on run month
`**PREDAY`	Selects records based on run date, less one (day)
`**PREWEK`	Selects records based on previous week (Sun–Sat)
`**PREMON`	Selects records based on previous month
`CURRENT`	Selects current period from the CIMSCalendar table
`PREVIOUS`	Selects previous period from the CIMSCalendar table

Examples:

```
DATE SELECTION 20050601 20050630
```

```
DATE SELECTION **PREMON
```

DEFAULT CLOSE DAY

Format:

```
DEFAULT CLODE DAY=nn
```

`nn` = 01–31

This statement overrides the system-wide close date set in the Configuration dialog box in CIMS Server Administrator. The year and month used for the close day reflect the year and month in which CIMSBill is run. For more information about the close date, see see *About the Close Date* on page 3-25.

Example:

```
DEFAULT CLOSE DAY=8
```

This statement sets the close date to the eighth of the month. If the CIMSBill run date is 20050706, the close date is set to 20050708. If the CIMSBill run date is 20050712, the close date is also set to 20050708.

DEFINE

This statement is used with the `CLIENT SEARCH ON` and `DYNAMIC CLIENT ADD ON` statements to specify the client account code structure. You can specify up to nine account levels (fields `J1–J9`).

Format:

```
DEFINE Jn start_loc len /desc/
```

`n` = 1–9

`start_loc` = starting position within the account code

`len` = total length of the level

`desc` = a description for the field

Example:

Account code `AAABBBCCDDDDD` contains four levels. To search from the lowest level to the highest level, use the following define fields:

```
DEFINE J1 1 3
DEFINE J2 1 6
DEFINE J3 1 8
DEFINE J4 1 13
CLIENT SEARCH ON
```

For the previous define fields:

```
J1=AAA
J2=AAABBB
J3=AAABBBCC
J4=AAABBBCCDDDDD
```

`CIMSBill` first searches for `AAABBBCCDDDDD` and if it does not find an entry it searches for `AAABBBCC` and so on.

DYNAMIC CLIENT ADD ON

Format:

```
DYNAMIC CLIENT ADD ON
```

This statement specifies that CIMSBill automatically insert corresponding client entries in the CIMSClient table for all clients that have a resource usage charge, but are not entered in the table. *Use this statement with caution because it can quickly fill the CIMS Client table with extraneous records*.

The inserted client entries will not include descriptions or contacts.

EXCLUDE

Format:

```
EXCLUDE,record_header,identifier_name,offset,length,low,high
```

`record_id` = record header

`identifier_name` = name of identifier

`offset` = starting offset into identifier value

`length` = identifier value length from offset (1–128)

`low` = low or from identifier value for comparison (1–128)

`high` = high or to identifier value for comparison (1–128)

When this statement is present, CIMSBill searches for records that contain the specified identifier name. Records in which the values for the identifier are equal to or greater than the low value and equal to or less than the high value are excluded from processing. All other records are included.

Example:

Assume that you want to exclude all UNIX storage records that contain `UserName` identifier values that begin with `geo`. The control statement would be:

```
EXCLUDE,UNIXSTOR,UserName,1,3,geo,geo
```

INCLUDE

Format:

```
INCLUDE,record_header,identifier_name,offset,length,low,high
```

`record_id` = record header

`identifier_name` = name of identifier

`offset` = starting offset into identifier value

`length` = identifier value length from offset (1–128)

`low` = low or from identifier value for comparison (1–128)

`high` = high or to identifier value for comparison (1–128)

When this statement is present, CIMSBill searches for records that contain the specified identifier name. Records in which the values for the identifier are equal to or greater than the low value and equal to or less than the high value are included in processing. All other records are excluded.

Example:

Assume that you want to include only those UNIX storage records that contain `UserName` identifier values that begin with `geo`. The control statement would be:

```
INCLUDE,UNIXSTOR,UserName,1,3,geo,geo
```

KEEP ORIGINAL CPU VALUES

Format:

```
KEEP ORIGINAL CPU VALUES
```

If CPU normalization is performed (see *NORMALIZE CPU VALUES*), this statement instructs CIMSBill to save the original CPU value as an identifier value. The identifier name is prefixed by `Orig_` followed by the rate code.

Example:

Assume that a CSR record contains the CPU rate code `LLA105` with a resource value of `1.15`. If CPU normalization is performed and the `KEEP ORIGINAL CPU VALUES` control statement is present, a new identifier will be built for the record with the identifier name `Orig_LLA105` and a value of `1.15`.

NORMALIZE CPU VALUES

Format:

```
NORMALIZE CPU VALUES
```

This statement instructs CIMSBill to normalize CPU values. For more information about normalizing CPU values, see page 3-41.

REPORT DATE

Important! • CIMS Lab recommends that you do not use this statement. This statement will place report dates rather than actual usage end dates in the accounting date fields of the CIMSBill Summary records.

Format:

```
REPORT DATE {YYYYYMMDD YYYYYMMDD | keyword}
```

This statement specifies the dates that are used as the accounting dates in the CIMS Summary records created by CIMSBill. This is the date that is used for reporting purposes in CIMS Server. You can use the following values:

- From and to dates

  or

- One of the following keywords:

Keyword	Description
`**RNDATE`	Sets the date range based on the run date
`**CURDAY`	Sets the date range based on the run date and the run date less one day
`**CURWEK`	Sets the date range based on the run week (Sun–Sat)
`**CURMON`	Sets the data range based on the run month
`**PREDAY`	Sets the date range based on the run date, less one day
`**PREWEK`	Sets the date range based on the previous week (Sun–Sat)
`**PREMON`	Sets the date range based on the previous month
`CURRENT`	Sets the date range based on the current period from the CIMSCalendar table
`PREVIOUS`	Sets the date range based on the previous period from the CIMSCalendar table
`USE DETAIL DATES`	Sets the date range based on the `CIMSAcct` Detail record (consult CIMS Lab before using this keyword)

Examples:

```
REPORT DATE 20050601 20050630
```

```
REPORT DATE **PREMON
```

USE SHIFT CODES

Format:

```
USE SHIFT CODES
```

This statement instructs CIMSBill to use the shift character specified in the CIMSAcct Output CSR+ record and apply the appropriate rate shift value.

If a shift value is not defined for the rate, the default rate value for the resource will be used. The default rate value is also the SHIFT 1 rate value.

B

Running the CIMS Server for UNIX Install Script

This chapter provides an example run of the `CS_cims_install` script. The various prompts and results from the expected replies are shown.

Example Install

```
********************************************************************************

Starting CIMS Server for Unix ./CS_cims_install Script at Thu Mar 31 10:45:34 CST 2005

********************************************************************************



This installation script automatically installs CIMS Server for Unix. You

will be prompted for various pieces of information throughout.

Default answers are denoted between square brackets ([]).  You may quit from

this script at any prompt which asks a "y"/"n" question.  CIMS Server for

Unix may partially install depending on the point at which you quit.


This script assumes that you have read the CIMS Server for Unix documentation

If you have not, it is recommended even if you are reinstalling or upgrading.


During this installation, you will be prompted for the following information


    - Location to install CIMS Server for Unix.

    - Path to the Java JRE bin directory.

    - If Perl is not accessable as /usr/bin/perl, you will

      be prompted if you wish to create a link from /usr/bin/perl

      to the location of perl on this machine. You will need to

      know the full path to the perl executable.


If you are not prepared to respond to these prompts, quit now...



Press enter when ready or "q" to quit:


Do you want CIMS Server for Unix installed in /usr/cims [y]?
```

```
Enter directory containing CIMS Server for Unix distribution files [/usr/cims/tmp]:



Provide the username and group of the account that is to maintain CIMS/UNIX Server.

The default is root and 0, respectively.  If you provide any other user and group,

the user account and group must be established.



Enter username of account that maintains CIMS/UNIX [root]: cims


Enter the group to which the username belongs [0]: cims


./CS_cims_install: Creating directory /usr/cims/log


./CS_cims_install: Copying files to /usr/cims directory ...


./CS_cims_install: Extracting CIMS/UNIX Server files ...

x bin

x bin/CIMSAdmin, 3004 bytes, 6 media blocks.

x collectors

x collectors/Universal

x collectors/Universal/Universal.pl, 15986 bytes, 32 media blocks.

x collectors/Apache

x collectors/Apache/Apache.pl, 15977 bytes, 32 media blocks.

x config

x config/CIMSDBConfig.xml, 335 bytes, 1 media blocks.

x jlib

x jlib/CIMSAcct.jar, 35191 bytes, 69 media blocks.

x jlib/CIMSAdmin.jar, 41044 bytes, 81 media blocks.

x jlib/CIMSBill.jar, 48446 bytes, 95 media blocks.

x jlib/CIMSLoad.jar, 36045 bytes, 71 media blocks.

x jlib/CIMSUtil.jar, 64478 bytes, 126 media blocks.
```

```
x lib

x lib/CIMS_lib

x lib/CIMS_lib/CS_libs.pm, 23100 bytes, 46 media blocks.

x lib/CIMS_lib/CS_aggtools.pm, 9976 bytes, 20 media blocks.

x log

x sample_processes

x sample_processes/UnixOS

x sample_processes/UnixOS/CurrentCSR.txt, 270247 bytes, 528 media blocks.

x sample_processes/UnixOS/JobUnixOS.pl, 8961 bytes, 18 media blocks.

x sample_processes/UnixOS/ProcCntl.xml, 1358 bytes, 3 media blocks.

x sample_processes/UnixOS/AcctTabl.txt, 303 bytes, 1 media blocks.

x sample_processes/UnixOS/sample

x sample_processes/UnixOS/sample/20050301.txt, 286250 bytes, 560 media blocks.

x sample_processes/UnixOS/sample/20050302.txt, 287253 bytes, 562 media blocks.

x sample_processes/UnixOS/sample/20050303.txt, 295015 bytes, 577 media blocks.

x sample_processes/UnixOS/sample/20050304.txt, 310175 bytes, 606 media blocks.

x sample_processes/UnixOS/sample/20050305.txt, 275570 bytes, 539 media blocks.

x sample_processes/UnixOS/sample/20050306.txt, 287691 bytes, 562 media blocks.

x sample_processes/UnixOS/sample/20050307.txt, 314348 bytes, 614 media blocks.

x sample_processes/UnixOS/sample/20050308.txt, 332735 bytes, 650 media blocks.

x sample_processes/UnixOS/sample/20050309.txt, 261789 bytes, 512 media blocks.

x sample_processes/UnixOS/sample/20050310.txt, 276869 bytes, 541 media blocks.

x sample_processes/UnixOS/sample/20050311.txt, 261940 bytes, 512 media blocks.

x sample_processes/UnixOS/sample/20050312.txt, 258477 bytes, 505 media blocks.

x sample_processes/UnixOS/sample/20050313.txt, 276803 bytes, 541 media blocks.

x sample_processes/UnixOS/sample/20050314.txt, 300416 bytes, 587 media blocks.

x sample_processes/UnixOS/sample/20050315.txt, 283390 bytes, 554 media blocks.

x sample_processes/UnixOS/sample/20050316.txt, 295014 bytes, 577 media blocks.

x sample_processes/UnixOS/sample/20050317.txt, 368425 bytes, 720 media blocks.

x sample_processes/UnixOS/sample/20050318.txt, 275912 bytes, 539 media blocks.

x sample_processes/UnixOS/sample/20050319.txt, 264205 bytes, 517 media blocks.

x sample_processes/UnixOS/sample/20050320.txt, 273912 bytes, 535 media blocks.
```

```
x sample_processes/UnixOS/sample/20050321.txt, 290314 bytes, 568 media blocks.
x sample_processes/UnixOS/sample/20050322.txt, 287023 bytes, 561 media blocks.
x sample_processes/UnixOS/sample/20050323.txt, 300189 bytes, 587 media blocks.
x sample_processes/UnixOS/sample/20050324.txt, 282335 bytes, 552 media blocks.
x sample_processes/UnixOS/sample/20050325.txt, 276161 bytes, 540 media blocks.
x sample_processes/UnixOS/sample/20050326.txt, 266604 bytes, 521 media blocks.
x sample_processes/UnixOS/sample/20050327.txt, 278430 bytes, 544 media blocks.
x sample_processes/UnixOS/sample/20050328.txt, 294446 bytes, 576 media blocks.
x sample_processes/UnixOS/sample/20050329.txt, 274916 bytes, 537 media blocks.
x sample_processes/UnixOS/sample/20050330.txt, 270247 bytes, 528 media blocks.
x sample_processes/UnixFS
x sample_processes/UnixFS/CurrentCSR.txt, 14248 bytes, 28 media blocks.
x sample_processes/UnixFS/JobUnixFS.pl, 8961 bytes, 18 media blocks.
x sample_processes/UnixFS/ProcCntl.xml, 1423 bytes, 3 media blocks.
x sample_processes/UnixFS/AcctTabl.txt, 379 bytes, 1 media blocks.
x sample_processes/UnixFS/sample
x sample_processes/UnixFS/sample/20050301.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050302.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050303.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050304.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050305.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050306.txt, 14598 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050307.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050308.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050309.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050310.txt, 13017 bytes, 26 media blocks.
x sample_processes/UnixFS/sample/20050311.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050312.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050313.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050314.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050315.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050316.txt, 14600 bytes, 29 media blocks.
```

```
x sample_processes/UnixFS/sample/20050317.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050318.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050319.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050320.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050321.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050322.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050323.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050324.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050325.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050326.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050327.txt, 14599 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050328.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050329.txt, 14600 bytes, 29 media blocks.
x sample_processes/UnixFS/sample/20050330.txt, 14248 bytes, 28 media blocks.
x sample_processes/UnixORA
x sample_processes/UnixORA/CurrentCSR.txt, 6410 bytes, 13 media blocks.
x sample_processes/UnixORA/JobUnixORA.pl, 8962 bytes, 18 media blocks.
x sample_processes/UnixORA/ProcCntl.xml, 1357 bytes, 3 media blocks.
x sample_processes/UnixORA/AcctTabl.txt, 303 bytes, 1 media blocks.
x sample_processes/UnixORA/sample
x sample_processes/UnixORA/sample/20050301.txt, 7342 bytes, 15 media blocks.
x sample_processes/UnixORA/sample/20050302.txt, 6725 bytes, 14 media blocks.
x sample_processes/UnixORA/sample/20050303.txt, 7603 bytes, 15 media blocks.
x sample_processes/UnixORA/sample/20050304.txt, 10103 bytes, 20 media blocks.
x sample_processes/UnixORA/sample/20050305.txt, 7354 bytes, 15 media blocks.
x sample_processes/UnixORA/sample/20050306.txt, 6736 bytes, 14 media blocks.
x sample_processes/UnixORA/sample/20050307.txt, 10701 bytes, 21 media blocks.
x sample_processes/UnixORA/sample/20050308.txt, 8600 bytes, 17 media blocks.
x sample_processes/UnixORA/sample/20050309.txt, 11008 bytes, 22 media blocks.
x sample_processes/UnixORA/sample/20050310.txt, 7293 bytes, 15 media blocks.
x sample_processes/UnixORA/sample/20050311.txt, 8500 bytes, 17 media blocks.
x sample_processes/UnixORA/sample/20050312.txt, 6732 bytes, 14 media blocks.
```

```
x sample_processes/UnixORA/sample/20050313.txt, 7010 bytes, 14 media blocks.

x sample_processes/UnixORA/sample/20050314.txt, 8228 bytes, 17 media blocks.

x sample_processes/UnixORA/sample/20050315.txt, 8219 bytes, 17 media blocks.

x sample_processes/UnixORA/sample/20050316.txt, 7302 bytes, 15 media blocks.

x sample_processes/UnixORA/sample/20050317.txt, 6703 bytes, 14 media blocks.

x sample_processes/UnixORA/sample/20050318.txt, 7006 bytes, 14 media blocks.

x sample_processes/UnixORA/sample/20050319.txt, 6401 bytes, 13 media blocks.

x sample_processes/UnixORA/sample/20050320.txt, 6389 bytes, 13 media blocks.

x sample_processes/UnixORA/sample/20050321.txt, 6404 bytes, 13 media blocks.

x sample_processes/UnixORA/sample/20050322.txt, 7313 bytes, 15 media blocks.

x sample_processes/UnixORA/sample/20050323.txt, 7596 bytes, 15 media blocks.

x sample_processes/UnixORA/sample/20050324.txt, 6989 bytes, 14 media blocks.

x sample_processes/UnixORA/sample/20050325.txt, 7290 bytes, 15 media blocks.

x sample_processes/UnixORA/sample/20050326.txt, 6407 bytes, 13 media blocks.

x sample_processes/UnixORA/sample/20050327.txt, 6094 bytes, 12 media blocks.

x sample_processes/UnixORA/sample/20050328.txt, 6694 bytes, 14 media blocks.

x sample_processes/UnixORA/sample/20050329.txt, 9458 bytes, 19 media blocks.

x sample_processes/UnixORA/sample/20050330.txt, 6410 bytes, 13 media blocks.

x sample_processes/UnixORAstorage

x sample_processes/UnixORAstorage/CurrentCSR.txt, 29740 bytes, 59 media blocks.

x sample_processes/UnixORAstorage/JobUnixORAstorage.pl, 8969 bytes, 18 media blocks.

x sample_processes/UnixORAstorage/ProcCntl.xml, 1346 bytes, 3 media blocks.

x sample_processes/UnixORAstorage/AcctTabl.txt, 303 bytes, 1 media blocks.

x sample_processes/UnixORAstorage/sample

x sample_processes/UnixORAstorage/sample/20050301.txt, 28602 bytes, 56 media blocks.

x sample_processes/UnixORAstorage/sample/20050302.txt, 28602 bytes, 56 media blocks.

x sample_processes/UnixORAstorage/sample/20050303.txt, 28602 bytes, 56 media blocks.

x sample_processes/UnixORAstorage/sample/20050304.txt, 28758 bytes, 57 media blocks.

x sample_processes/UnixORAstorage/sample/20050305.txt, 28757 bytes, 57 media blocks.

x sample_processes/UnixORAstorage/sample/20050306.txt, 28756 bytes, 57 media blocks.

x sample_processes/UnixORAstorage/sample/20050307.txt, 29086 bytes, 57 media blocks.

x sample_processes/UnixORAstorage/sample/20050308.txt, 29086 bytes, 57 media blocks.
```

```
x sample_processes/UnixORAstorage/sample/20050309.txt, 29736 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050310.txt, 19751 bytes, 39 media blocks.
x sample_processes/UnixORAstorage/sample/20050311.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050312.txt, 29736 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050313.txt, 29736 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050314.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050315.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050316.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050317.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050318.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050319.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050320.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050321.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050322.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050323.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050324.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050325.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050326.txt, 29736 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050327.txt, 29736 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050328.txt, 29737 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050329.txt, 29740 bytes, 59 media blocks.
x sample_processes/UnixORAstorage/sample/20050330.txt, 29740 bytes, 59 media blocks.
x sample_processes/UnixDB2
x sample_processes/UnixDB2/CurrentCSR.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/JobUnixDB2.pl, 8962 bytes, 18 media blocks.
x sample_processes/UnixDB2/ProcCntl.xml, 1347 bytes, 3 media blocks.
x sample_processes/UnixDB2/AcctTabl.txt, 303 bytes, 1 media blocks.
x sample_processes/UnixDB2/sample
x sample_processes/UnixDB2/sample/20050301.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050302.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050303.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050304.txt, 0 bytes, 0 media blocks.
```

```
x sample_processes/UnixDB2/sample/20050305.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050306.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050307.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050308.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050309.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050310.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050311.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050312.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050313.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050314.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050315.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050316.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050317.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050318.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050319.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050320.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050321.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050322.txt, 417 bytes, 1 media blocks.
x sample_processes/UnixDB2/sample/20050323.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050324.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050325.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050326.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050327.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050328.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050329.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2/sample/20050330.txt, 0 bytes, 0 media blocks.
x sample_processes/UnixDB2storage
x sample_processes/UnixDB2storage/CurrentCSR.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/JobUnixDB2storage.pl, 8969 bytes, 18 media blocks.
x sample_processes/UnixDB2storage/ProcCntl.xml, 1347 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/AcctTabl.txt, 303 bytes, 1 media blocks.
x sample_processes/UnixDB2storage/sample
```

```
x sample_processes/UnixDB2storage/sample/20050301.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050302.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050303.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050304.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050305.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050306.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050307.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050308.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050309.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050310.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050311.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050312.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050313.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050314.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050315.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050316.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050317.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050318.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050319.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050320.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050321.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050322.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050323.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050324.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050325.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050326.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050327.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050328.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050329.txt, 1376 bytes, 3 media blocks.
x sample_processes/UnixDB2storage/sample/20050330.txt, 1376 bytes, 3 media blocks.
x sample_processes/Apache
x sample_processes/Apache/CurrentCSR.txt, 429 bytes, 1 media blocks.
```

```
x sample_processes/Apache/JobApache.pl, 8961 bytes, 18 media blocks.

x sample_processes/Apache/ProcCntl.xml, 1184 bytes, 3 media blocks.

x sample_processes/Apache/AcctTabl.txt, 265 bytes, 1 media blocks.

x sample_processes/Apache/sample

x sample_processes/Apache/sample/20050302.txt, 7800 bytes, 16 media blocks.

x sample_processes/Apache/sample/20050303.txt, 648 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050304.txt, 3931 bytes, 8 media blocks.

x sample_processes/Apache/sample/20050305.txt, 2772 bytes, 6 media blocks.

x sample_processes/Apache/sample/20050306.txt, 1979 bytes, 4 media blocks.

x sample_processes/Apache/sample/20050308.txt, 3952 bytes, 8 media blocks.

x sample_processes/Apache/sample/20050309.txt, 1525 bytes, 3 media blocks.

x sample_processes/Apache/sample/20050310.txt, 1495 bytes, 3 media blocks.

x sample_processes/Apache/sample/20050311.txt, 863 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050312.txt, 859 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050315.txt, 636 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050316.txt, 429 bytes, 1 media blocks.

x sample_processes/Apache/sample/20050317.txt, 664 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050322.txt, 666 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050324.txt, 666 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050325.txt, 192 bytes, 1 media blocks.

x sample_processes/Apache/sample/20050326.txt, 192 bytes, 1 media blocks.

x sample_processes/Apache/sample/20050328.txt, 1310 bytes, 3 media blocks.

x sample_processes/Apache/sample/20050329.txt, 873 bytes, 2 media blocks.

x sample_processes/Apache/sample/20050330.txt, 429 bytes, 1 media blocks.

x scripts

x scripts/sample_nightly.pl, 5232 bytes, 11 media blocks.

x scripts/sample_nightly.sh, 2146 bytes, 5 media blocks.

x scripts/sample_procCIMS.pl, 7208 bytes, 15 media blocks.


./CS_cims_install: Creating the CIMS Server for Unix Configuration file, /etc/cims.conf


./CS_cims_install: Setting file permissions/ownership for binaries and scripts ...
```

```
./CS_cims_install: Creating processes directory from sample_processes


Enter Java JRE bin directory path [/opt/java1.4/jre/bin/]: /usr/java14/jre/bin



* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                              *
*      YOU HAVE SUCCESSFULLY INSTALLED CIMS SERVER FOR UNIX                    *
*                                                                              *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *





********************************************************************************
Ending CIMS Server for Unix ./CS_cims_install Script at Thu Mar 31 11:02:09 CST 2005
********************************************************************************
```

Glossary

CIMS Server Resource (CSR) File • The resource file that contains the data that is input into CIMS Server for UNIX. The CIMS Server Resource file contains CIMS Server Resource records. These records are comma-delimited and can contain a very large number of resource identifiers and resources. *See also identifier and rate code.*

CIMS Server Resource Plus (CSR+) File • These files are produced by CIMS Mainframe 12.0 and later. CSR+ files are similar to CIMS Server Resource (CSR) files, with the exception that the records in the CSR+ file contain an additional header at the beginning of the record.

client • **1**. A computer that accesses shared network resources provided by a server. **2.** In CIMS Server, an entity that consumes resources. A client is represented by an account code.

crontab • A UNIX command that creates a table or list of commands, each of which is to be executed by the operating system at a specified time.

COM • Acronym for Component Object Model. A specification developed by Microsoft for building software components that can be assembled into programs or add functionality to existing programs running on Microsoft Windows platforms.

driver • A program that interacts with a particular device or type of software. The driver contains the special knowledge of the device or software interface that programs using the driver do not.

identifier • In the CIMS Server Resource record, a unique key that denotes the source of a resource that has been consumed. Examples include device name, server name, system ID, phone number, user ID, state code or building number. A consumed resource can have one to many identifiers.

JAR file • A file that contains the class, image, and sound files for a Java application gathered into a single file and compressed.

Java • An object-oriented programming language developed by Sun Microsystems. Java is designed to be platform neutral.

JDBC • An application program interface (API) specification for connecting programs written in Java to the data in popular databases.

JRE • Acronym for Java Runtime Environment. The JRE is part of the Java Development Kit (JDK), a set of programming tools for developing Java applications. The JRE provides the minimum requirements for executing a Java application; it consists of the Java Virtual Machine (JVM), core classes, and supporting files.

ODBC • Acronym for Open Database Connectivity. An interface providing a common language for database access.

offset • A number that indicates how far from a starting point an item is located.

Perl • Acronym for Practical Extraction and Report Language. A script programming language that is similar in syntax to the C language and that includes a number of popular UNIX facilities such as SED, awk, and tr.

rate code • A rate code represents the resource units being reported (for example, CPU time, transactions processed, or lines printed). The rate code includes the value for the resource and other rate processing information.

XML • Acronym for Extensible Markup Language. A simple, very flexible text format derived from SGML. XML allows for more precise declarations or content and more meaningful search results across multiple platforms.

Index