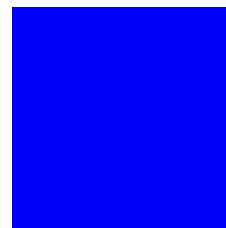


# **CIMS Lab, Inc.**

---



## **CIMS Data Collectors**

### **for Microsoft® Windows®**

## **Installation and User Guide**

Version 4.2

CIMS Publication Number: DC-UG-420-00

Published 10/03/05

Information in this guide is subject to change without notice and does not constitute a commitment on the part of CIMS Lab, Inc. It is supplied on an "as is" basis without any warranty of any kind, either explicit or implied. Information may be changed or updated in this guide at any time.

### **Copyright Information**

CIMS is ©copyright 1974 - 2005 by CIMS Lab, Inc. and its subsidiaries. This guide is ©copyright 1974 - 2005 by CIMS Lab, Inc. and its subsidiaries and may not be reproduced in whole or in part, by any means, without the written permission of CIMS Lab, Inc. and its subsidiaries.

Names marked ™ or ® and other company and product names may be trademarks or registered trademarks of their respective vendors or organizations.

### **Mailing Address**

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

# Table of Contents



## Preface

<b>About CIMS Lab</b> .....	<b>xi</b>
<b>Contacting CIMS Lab</b> .....	<b>xii</b>
<b>About This Guide</b> .....	<b>xii</b>
How to Use This Guide .....	xiii
Document Conventions .....	xiv
Terminology Used in this Guide .....	xiv
<b>Related Publications</b> .....	<b>xv</b>

## 1 • About CIMS Data Collectors

<b>What are CIMS Data Collectors?</b> .....	<b>1-2</b>
<b>About CIMS Data Collectors for Microsoft Windows</b> .....	<b>1-3</b>
CIMS Application-specific Data Collectors .....	1-4
CIMS Universal Data Collector .....	1-5
<b>How CIMS Data Collectors for Microsoft Windows Integrate With CIMS Chargeback Systems</b>	<b>1-6</b>

## 2 • Installing CIMS Data Collectors and Setting Up the System

<b>System Specifications</b> .....	<b>2-2</b>
<b>Installing CIMS Data Collectors</b> .....	<b>2-2</b>
<b>CIMS Data Collectors Architecture</b> .....	<b>2-3</b>
CIMS Job Runner Program .....	2-3
Job Files (JobFiles Folder) .....	2-5
Job File XML Schema .....	2-5
Collection Files (Collectors Folder) .....	2-6
Job Log Files (LogFiles Folder) .....	2-8
CIMS Processing Programs (Process Engine\JobLibrary Folder) .....	2-9
Process Definitions (Processes Folder) .....	2-12
Scripts (Scripts Folder) .....	2-14

<b>Setting Up Proration Files (Optional)</b> .....	<b>2-15</b>
Creating a Proration Table .....	2-16
Creating the CIMSPrat Parameters File .....	2-17
Proration Example .....	2-21
<b>Setting Up the System</b> .....	<b>2-25</b>
Creating Job Files .....	2-25
<b>Running CIMS Data Collectors</b> .....	<b>2-92</b>
Data Processing Frequency .....	2-92
Required Folder Permissions for Data Processing .....	2-92
Running CIMS Job Runner .....	2-93

### **3 • Operating System Data Collectors**

<b>Windows Process Data Collector</b> .....	<b>3-2</b>
Creating a Log On User Account for the CIMS Windows Process Collector Service (Optional) ..	3-2
System Configuration Options for the CIMS Windows Process Collector .....	3-4
Installing the CIMS Windows Process Collector .....	3-9
Enabling CIMS Windows Process Logging .....	3-11
CIMS Windows Process Collector Log File Format .....	3-13
Identifiers and Resources Collected From the CIMS Windows Process Collector Log File .....	3-18
Setting Up the CIMS Windows Process Collector .....	3-19
Running the CIMS Windows Process Collector .....	3-24
<b>Windows System Resource Manager (WSRM) Collector</b> .....	<b>3-25</b>
Identifiers and Resources Collected by the WSRM Collector .....	3-25
Setting Up the WSRM Collector .....	3-26
<b>Citrix Data Collector</b> .....	<b>3-28</b>
Identifiers and Resources Collected by the Citrix Collector .....	3-28
Setting Up the Citrix Collector .....	3-29
Running the Citrix Collector .....	3-31
<b>VMware Data Collector</b> .....	<b>3-31</b>
Identifiers and Resources Collected by the VMware Collector .....	3-31
Setting Up the VMware Collector .....	3-32
Running the VMware Collector .....	3-34
<b>AS/400 Data Collector</b> .....	<b>3-34</b>

## 4 • Database Data Collectors

<b>Microsoft SQL Server 2000 Data Collector</b> .....	<b>4-3</b>
Enabling SQL Server 2000 Tracing .....	4-4
SQL Server 2000 Trace File Format .....	4-6
Identifiers and Resources Collected from the SQL Server 2000 Trace File .....	4-8
Setting Up the SQL Server 2000 Collector .....	4-9
Running the SQL Server 2000 Collector .....	4-17
<b>Oracle Data Collector</b> .....	<b>4-18</b>
Setting Up the CIMSWIND Collector .....	4-18
Creating a Process Definition Folder for Oracle Data Collection .....	4-21
Enabling Oracle Logging .....	4-21
Resources Collected .....	4-23
Setting Up the Oracle Collector Job File .....	4-24
Running the Oracle Collector .....	4-25
<b>DB2 Data Collector</b> .....	<b>4-26</b>
Setting up the CIMSWIND Collector .....	4-26
Creating a Process Definition Folder for DB2 Data Collection .....	4-26
Enabling DB2 Logging .....	4-27
Resources Collected .....	4-29
Setting Up the DB2 Collector Job File .....	4-31
Running the DB2 Collector .....	4-32
<b>Sybase Data Collector</b> .....	<b>4-33</b>
<b>Database Size Data Collector (DBSpace)</b> .....	<b>4-33</b>
Identifiers and Resources Collected by the DBSpace Collector .....	4-33
Setting Up the DBSpace Collector .....	4-33
Running the DBSpace Collector .....	4-36
<b>SQL Server Reporting Services Data Collector</b> .....	<b>4-37</b>
Identifiers and Resources Collected by the Reporting Services Collector .....	4-37
Setting Up the Reporting Services Collector .....	4-38

**5 • E-mail Data Collectors**

**Microsoft Exchange Server 5.5 Collector** .....5-2  
Enabling Exchange Server 5.5 Logging ..... 5-2  
Exchange Server 5.5 Log File Name and Format ..... 5-3  
Identifiers and Resources Collected from the Exchange Server 5.5 Log File ..... 5-5

**Microsoft Exchange Server 2000/2003 Collectors** .....5-6  
Enabling Exchange Server 2000/2003 Logging ..... 5-6  
Exchange Server 2000/2003 Log File Name and Format ..... 5-7  
Identifiers and Resources Collected from the  
Exchange Server 2000/2003 Log File ..... 5-9

**Setting Up and Running the Exchange Server Collectors** ..... 5-12  
Setting Up the Exchange Server Collectors ..... 5-12  
Running the Exchange Server Collectors ..... 5-14

**Microsoft Exchange Server Mailbox 5.5, 2000, and 2003 Data Collector** ..... 5-15  
Requirements ..... 5-15  
Troubleshooting ..... 5-16  
Identifiers and Resources Collected from the  
Exchange Server Mailbox Store ..... 5-17  
Setting Up the Exchange Server Mailbox Collector ..... 5-20  
Running the Exchange Server Mailbox Collector ..... 5-23

**Microsoft Outlook Web Access Data Collection** ..... 5-24

**Lotus Notes Data Collector** ..... 5-24

**6 • Internet Data Collectors**

**Microsoft Internet Information Services (IIS)  
Data Collector** .....6-3  
Enabling IIS Logging ..... 6-3  
IIS Log File Format ..... 6-4  
Identifiers and Resources Collected from the IIS Log File ..... 6-6  
Setting Up the IIS Collector ..... 6-8  
Running the IIS Collector ..... 6-10

**Microsoft Internet Security and Acceleration (ISA) Server Data Collector** ..... 6-11  
Enabling ISA Logging ..... 6-11  
ISA Server Log File Format ..... 6-12  
Identifiers and Resources Collected from the ISA Server Log File ..... 6-16  
Setting Up the ISA Server Collector ..... 6-17  
Running the ISA Server Collector ..... 6-18

<b>Microsoft Proxy Server Data Collector</b> .....	<b>6-19</b>
Enabling Proxy Server Logging .....	6-19
Proxy Server Log File Format .....	6-20
Identifiers and Resources Collected from the Proxy Server Log File .....	6-24
Setting Up the Proxy Server Collector .....	6-25
Running the Proxy Server Collector .....	6-26
<b>SQUID Data Collector</b> .....	<b>6-27</b>
Identifiers and Resources Collected from the SQUID Log File .....	6-27
Setting Up the SQUID Collector .....	6-27
Running the SQUID Collector .....	6-29
<b>Sendmail Data Collector</b> .....	<b>6-30</b>
Identifiers and Resources Collected from the Sendmail Log File .....	6-30
Setting Up the Sendmail Collector .....	6-30
Running the Sendmail Collector .....	6-32
<b>Apache Data Collector</b> .....	<b>6-33</b>
Identifiers and Resources Collected from the Apache Log File .....	6-33
Setting Up the Apache Collector .....	6-33
Running the Apache Collector .....	6-35
<b>Netscape Proxy Server Data Collector</b> .....	<b>6-35</b>
<b>7 • Storage Data Collectors</b>	
<b>Windows Disk Data Collector</b> .....	<b>7-2</b>
Identifiers and Resources Collected by the CIMS Windows Disk Collector .....	7-2
Setting Up the CIMS Windows Disk Collector .....	7-3
Running the CIMS Windows Disk Collector .....	7-8
<b>Veritas NetBackup</b> .....	<b>7-9</b>
NetBackup Log File Format .....	7-9
Identifiers and Resources Collected from the NetBackup Log File .....	7-10
Setting Up the NetBackup Collector .....	7-11
Running the NetBackup Collector .....	7-12
<b>Veritas</b> .....	<b>7-12</b>

**8 • Network Data Collectors**

**NetFlow Data Collector** .....8-2  
Identifiers and Resources Collected from the  
NetFlow Data File ..... 8-2  
Setting Up the NetFlow Collector ..... 8-3  
Running the Netflow Collector ..... 8-4  
**Novell NetWare Data Collection** .....8-5  
**BindView Data Collector** .....8-5

**9 • Printer Data Collectors**

**Windows Event Log Data Collector for Print** .....9-2  
Identifiers and Resources Collected by the  
CIMS Windows Event Log Collector ..... 9-2  
Setting Up the CIMS Windows Event Log Collector ..... 9-3  
Setting Event Log Security ..... 9-8  
Running the CIMS Windows Event Log Collector ..... 9-8  
Setting the Event Viewer Options for the System Event Log ..... 9-9  
**Windows Print Data Collector** ..... 9-10  
Creating a Log On User Account for the CIMS Windows Print Collector Service (Optional) .... 9-10  
System Configuration Options for the CIMS Windows Print Collector ..... 9-11  
Installing the CIMS Windows Print Collector ..... 9-14  
Enabling Windows Print Logging ..... 9-16  
CIMS Windows Print Collector Log File Format ..... 9-17  
Identifiers and Resources Collected from the CIMS Windows Print Collector Log File ..... 9-18  
Setting Up the CIMS Windows Print Collector ..... 9-20  
Running the CIMS Windows Print Collector ..... 9-25

**10 • Transactions Collector**

**About Transactions** ..... 10-2  
**About the CIMSTransaction Table** ..... 10-2  
Identifiers and Resources Collected from the CIMSTransaction Table ..... 10-4  
**Setting Up the Transactions Collector** ..... 10-4  
**Running the Transactions Collector** ..... 10-6



<b>11 • Mainframe Data Collectors</b>	
About Mainframe Data Collection .....	11-2
Setting Up Collection for CSR or CSR+ Files .....	11-3
Setting Up Collection for CIMS Ident, Detail, and Summary Files .....	11-5
Running the Mainframe Collection Process .....	11-6
<b>12 • UNIX Data Collectors</b>	
About UNIX Data Collection .....	12-2
Setting Up Collection for CSR Files .....	12-2
Running the UNIX Collection Process .....	12-4
<b>13 • Other Data Collectors</b>	
SAP, Shiva, and Evolve Data Collectors .....	13-1
<b>14 • CIMS Universal Data Collector</b>	
About CIMS Universal Data Collector .....	14-2
The Data Conversion Process .....	14-2
<b>Creating a Conversion Definition Using CIMS Conversion Builder</b> .....	14-3
Creating a Definition file .....	14-3
Opening a Conversion Definition .....	14-24
Saving a Conversion Definition .....	14-24
Viewing Conversion Definitions .....	14-24
Running CIMS Conversion Engine .....	14-24
<b>Setting Up and Running the Universal Collector</b> .....	14-25
Adding Resource Rate Codes to the CIMSRate Table .....	14-25
Setting Up the Universal Collector .....	14-25
Running the Universal Collector .....	14-27
<b>Example Files</b> .....	14-28
Log File—SodaLog.txt .....	14-28
Conversion Definition File—SodaLogDef.txt .....	14-29
Output File—CurrentCSR.txt .....	14-37
<b>15 • Contacting Technical Support</b>	

**A • CIMS Aggregation Engine API**

**About CIMS Aggregation Engine** .....A-2  
About the Aggregation Algorithm ..... A-2  
Processing the Log File ..... A-3  
Processing the Work File ..... A-3  
**CIMSAggregation Interfaces** .....A-3  
**TypedEngine and ScriptingEngine Interfaces** .....A-4  
About Specifying Dates and Times ..... A-4  
TypedEngine and ScriptingEngine Interface Properties ..... A-5  
TypedEngine and ScriptingEngine Interface Methods ..... A-12  
**ExceptionFile Interface** ..... A-17  
ExceptionFile Interface Properties ..... A-17  
ExceptionFileInterface Method ..... A-18

**B • CIMS Data Sources**

**About CIMS Data Sources** .....B-2  
**Creating a CIMS Data Source** .....B-3

**Glossary**

**Index**



---

# Preface

As companies continue to integrate computer technology into their business operations, it becomes increasingly important to properly administer the IT function, particularly with respect to performance and cost.

CIMS Data Collectors enable you to collect reliable and useful data related to how your technology resources are being used. CIMS Data Collectors integrate with CIMS to enable you to view IT resource consumption within your enterprise and to fairly and accurately allocate costs.

The technology behind CIMS Data Collectors 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 provides instructions for installing, setting up, and running CIMS Data Collectors for Microsoft Windows. Because of its technical content, this guide is primarily intended for users that have experience working with the following Microsoft and other technologies:

- .NET Framework 1.1
- Windows Script Host (WSH)
- Visual Basic Script (VBScript)
- Component Object Model (COM)
- Extensible Markup Language (XML)

This guide assumes that users are familiar with concepts associated with the CIMS Server application including its architecture and functions and the layout and use of the CIMS Server Resource (CSR) and CIMS Server Resource Plus (CSR+) files. For more information about CIMS Server, including the layout of the CSR and CSR+ files, refer to the *CIMS Server Administrator's Guide*.

## How to Use This Guide

The following table describes the chapters in this guide. You should begin with *Chapter 2, Installing CIMS Data Collectors and Setting Up the System* and then continue to the collector-specific information provided in *Chapter 3* through *Chapter 14*.

Ch. No.	Chapter Name	Content Description
1	<i>About CIMS Data Collectors</i>	Provides an introduction to CIMS Data Collectors.
2	<i>Installing CIMS Data Collectors and Setting Up the System</i>	Provides steps for installing and setting up CIMS Data Collectors and an overview of the system architecture.
3	<i>Operating System Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for operating systems.
4	<i>Database Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for databases.
5	<i>E-mail Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for e-mail applications.
6	<i>Internet Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for internet applications.
7	<i>Storage Data Collectors</i>	Provides setup procedures for CIMS Data Collectors for storage systems.
8	<i>Network Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for network applications.
9	<i>Printer Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for printers.
10	<i>Transactions Collector</i>	Provides setup procedures for the CIMS Data Collector for transactions.
11	<i>Mainframe Data Collectors</i>	Provides setup procedures for processing CIMS Server Resource Plus (CSR+) files created by the CIMS Mainframe Data Collector and Chargeback System application.
12	<i>UNIX Data Collectors</i>	Provides setup procedures for processing CIMS Server Resource (CSR) files created by the CIMS Data Collector for UNIX application.
13	<i>Other Data Collectors</i>	Provides logging and setup procedures for CIMS Data Collectors for miscellaneous applications.
14	<i>CIMS Universal Data Collector</i>	Provides procedures for using CIMS Universal Data Collector.

Ch. No.	Chapter Name	Content Description
15	<i>Contacting Technical Support</i>	Provides technical support information.
A	<i>CIMS Aggregation Engine API</i>	Describes the CIMS Aggregation Engine API.

## Document Conventions

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

Symbol or Type Style	Represents	Example
<u>Alternate color</u>	hyperlinked cross-references to other sections in this guide; if you are viewing this guide online, you can click the cross-reference to jump directly to its location	.....see <a href="#">Data Migration</a> .
<i>Italic</i>	words that are emphasized	...the entry <i>after</i> the current entry...
	a new term	...by <i>identifier</i> values.
	the titles of other manuals	<i>CIMS Server Administrator's Guide</i>
	variables in file names	CIMSProcessLog-yyyymmdd.txt
<b>Bold</b>	names of interface items such as tabs, boxes, buttons, lists, and check boxes.	Select the <b>Use Local Time</b> check box Enter the path in the <b>Log File Path</b> box
Monospace	directories, file names, command names, computer code, computer screen text, system responses, command line commands, what the user types	Processes <b>folder</b> MSSQL2000.wsf <b>script</b>
< >	the name of a key on the keyboard	Press <Enter>
▶	navigating a menu or a folder	File ▶ Import ▶ Object

## Terminology Used in this Guide

For simplicity, in this guide, the term “application” refers to both applications and systems.

## Related Publications

As you use this guide, you might find it helpful to have these additional guides available for reference:

- *CIMS Server Administrator's Guide*
- *CIMS Desktop User Guide*
- *CIMS Mainframe Data Collector and Chargeback System User Guide*
- *CIMS Data Collector for UNIX User Guide*





---

# About CIMS Data Collectors

<b>What are CIMS Data Collectors?</b> .....	1-2
<b>About CIMS Data Collectors for Microsoft Windows</b> .....	1-3
CIMS Application-specific Data Collectors .....	1-4
CIMS Universal Data Collector .....	1-5
<b>How CIMS Data Collectors for Microsoft Windows Integrate With CIMS Chargeback Systems</b> .....	1-6

## What are CIMS Data Collectors?

CIMS Data Collectors read usage metering data generated by applications (usually standard usage metering files such as log files) and produce a common output file that integrates with the CIMS cost allocation and chargeback system. (See [How CIMS Data Collectors for Microsoft Windows Integrate With CIMS Chargeback Systems](#) on page 1-6).

CIMS Data Collectors are non-intrusive and do not affect system performance or operation. Most collectors gather data from files that are produced by an application's built-in usage metering functionality.

CIMS Data Collectors are available for the following platforms: Microsoft® Windows®, Unix/Linux, and Mainframe Systems.

This guide describes the data collectors that run on the Windows operating system. All references to CIMS Data Collectors in the following chapters refer to the collectors for the Windows system.

For information about data collection for other platforms, refer to the CIMS Mainframe Data Collector and Chargeback System and CIMS Data Collector for UNIX documentation.

## About CIMS Data Collectors for Microsoft Windows

CIMS Data Collectors for Microsoft Windows run on Microsoft 2000 Server or later operating system and collect data from Windows and Windows-compatible applications (including older Windows operating systems such as Windows NT) and non-Windows applications and operating systems.

CIMS Lab provides the following types of Windows data collectors:

- Application-specific collectors for commonly used Windows and Windows-compatible applications. These collectors gather and convert usage data into a file format, the CIMS Server Resource (CSR) file, that can be processed by CIMS Server or CIMS Mainframe Data Collector and Chargeback System.

---

**Note** • This guide provides the steps required to process CSR file using CIMS Server. To process CSR files using the CIMS Mainframe system, refer to the *CIMS Mainframe Data Collector and Chargeback System User Guide*.

---

Depending on the type of data to be collected, these collectors gather data from usage metering files or directly from the application or system.

- **From usage metering files.** Usage metering files are either produced by the collector itself or produced by an application's built-in usage metering functionality.

Examples of collectors that create usage metering files are the CIMS Windows Process and CIMS Windows Print collectors, which produce log files. These "logger" collectors are built by CIMS Lab.

Examples of collectors that use an application's usage metering file are the Microsoft IIS, Exchange Server, and SQL Server data collectors.

- **From the application or system.** These collectors (referred to as "snapshot" collectors) do not require a usage metering file. They collect data directly from the application or system. The data collected is current as of the date and time that the collector is run.

Examples of collectors that collect data directly from an application or system are the CIMS Windows Disk Collector, which collects disk storage data, and the DBSpace collector, which collects size data for SQL Server and/or Sybase databases.

- A universal collector, CIMS Universal Collector, for applications that do not have a specific collector. CIMS Universal Collector allows for the collection of *any usage metering data from any application*. This collector also produces a CSR file that can be processed by CIMS Server or CIMS Mainframe Data Collector and Chargeback System.
- Collectors that process the files produced by the CIMS UNIX and Mainframe collectors for input into CIMS Server.

The following sections list the applications that are supported by CIMS Data Collectors for Microsoft Windows.

## CIMS Application-specific Data Collectors

Individual data collectors are available for the following applications. For applications supported by CIMS Universal Collector, see *CIMS Universal Data Collector* on page 1-5.

<b>Operating Systems</b>	Microsoft Windows NT 4.0, Microsoft Windows 2000/2003/XP, z/OS, UNIX, Citrix, VMware, AS/400
<b>Databases</b>	Microsoft SQL Server, Oracle, DB2, Sybase
<b>Internet Applications</b>	Microsoft Internet Information Services (IIS), Microsoft ISA/Proxy Server, SQUID, Apache, sendmail, Netscape Proxy Server
<b>E-mail Applications</b>	Microsoft Exchange Server, Microsoft Exchange Server Mailbox, Microsoft Outlook Web Access, Lotus Notes
<b>Storage Systems</b>	Windows File Systems (NTFS), Veritas
<b>Network Applications</b>	CISCO Netflow, Novell Netware, BindView
<b>Others</b>	Printers, SAP, Shiva, Evolve

**Table 1-1 • CIMS Application-specific Data Collectors**

## CIMS Universal Data Collector

The following table lists some (not all) of the applications supported by CIMS Universal Data Collector. For information regarding applications not contained in this list or the list of application-specific collectors on page [page 1-4](#), contact CIMS Lab.

<b>Operating Systems</b>	OS/400, AIX, Solaris, Unisys, HP/UX, Netware, VM/VSE, etc.
<b>Databases</b>	IMS, IDMS, ADABAS, Focus, Datacom, Supra, M204, Informix, etc.
<b>Internet/Telecom Applications</b>	IBM WebSphere, BEA WebLogic, 3COM Routers, Firewalls, Proxy Servers, SurfWatch, Switches/Lines, PBX Systems, RMON2, etc.
<b>E-mail Applications</b>	Lotus Notes, AOL, MSN, GroupWare, Apache, Profs, etc.
<b>Storage Systems</b>	SANS, Backup Systems, Storage Monitors, Storage Managers, Tape Systems, Robots, DVD/CD, etc.
<b>ERP Applications</b>	PeopleSoft, Oracle Financials, Hyperion, JD Edwards, Lawson, BAAN, Walker, etc.
<b>Human Resource Applications</b>	WSG Empire Time, ChangePoint, TimeSlips, MS Project, Help Desks, Consultants, etc.
<b>Output Systems</b>	SAR/Express, CA/Dispatch, BUNDL, Infopac, RMS, Control-D, Print Servers and/or RDMS Systems, Printer Accounting Server, etc.
<b>Other Applications</b>	BMC Patrol Suite, Candle, HP OpenView, ASG TMON, IBM Tivoli, CA Unicenter, Net IQ, WebTrends, CRM Products, etc.

**Table 1-2 • CIMS Universal Data Collector**

## How CIMS Data Collectors for Microsoft Windows Integrate With CIMS Chargeback Systems

CIMS Data Collectors for Microsoft Windows work with the CIMS Server and CIMS Mainframe chargeback systems. These chargeback systems use the data provided by the CIMS Data Collectors to track usage associated with databases, software packages, in-house applications, servers and workstations, and other systems. CIMS then accurately displays the resources used and the associated charges.

As shown in [Figures 1-1](#), it is useful to think of CIMS as a funnel that accepts usage data and returns organized information. This data is organized and restructured as a multitude of chargeback and management reports that can help IT managers and staff to track and allocate resources.

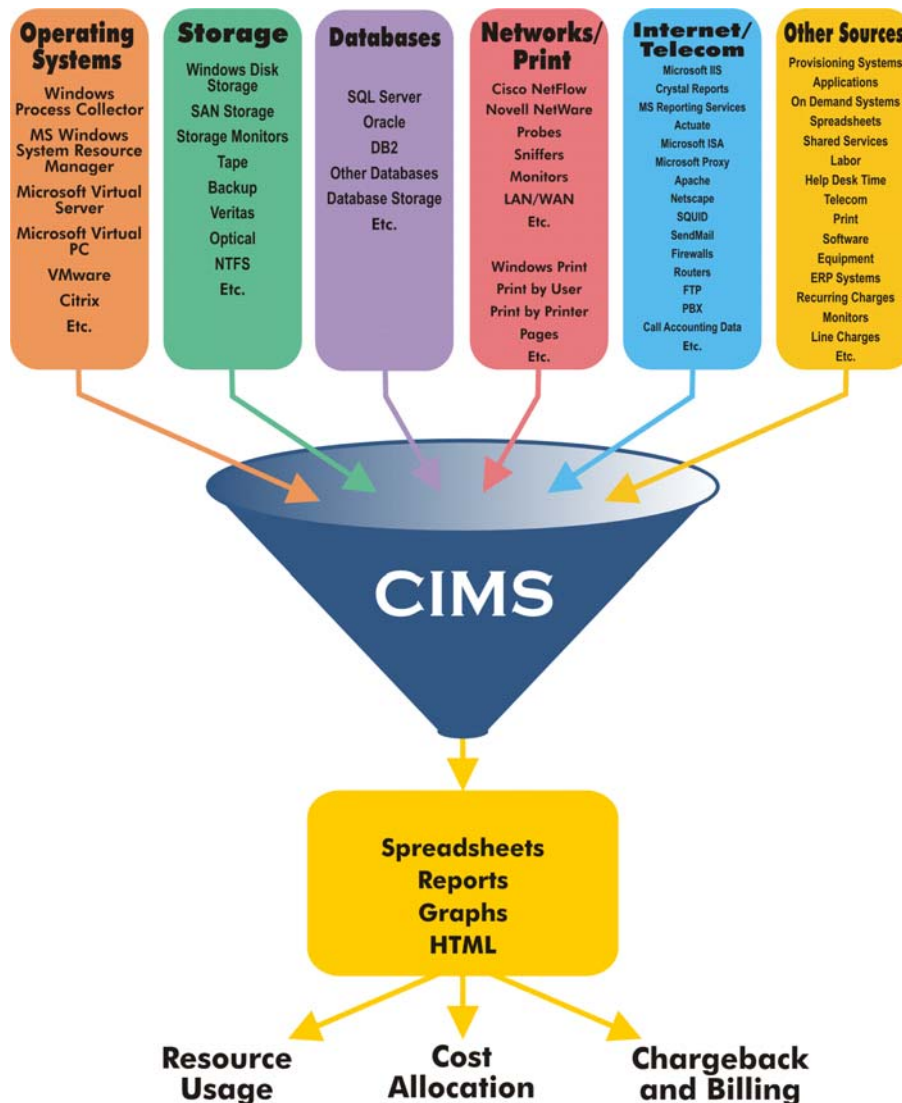


Figure 1-1 • CIMS collects usage data and organizes it as reporting information

# Installing CIMS Data Collectors and Setting Up the System

This chapter provides the installation and configuration instructions for CIMS Data Collectors. You should review this chapter before continuing to the collector-specific chapters in this guide.

- System Specifications** .....2-2
- Installing CIMS Data Collectors** .....2-2
- CIMS Data Collectors Architecture** .....2-3
  - CIMS Job Runner Program ..... 2-3
  - Job Files (JobFiles Folder) ..... 2-5
  - Job File XML Schema ..... 2-5
  - Collection Files (Collectors Folder) ..... 2-6
  - Job Log Files (LogFiles Folder) ..... 2-8
  - CIMS Processing Programs (Process Engine\JobLibrary Folder) ..... 2-9
  - Process Definitions (Processes Folder) ..... 2-12
  - Scripts (Scripts Folder) ..... 2-14
- Setting Up Proration Files (Optional)** .....2-15
  - Creating a Proration Table .....2-16
  - Creating the CIMSPrat Parameters File ..... 2-17
  - Proration Example ..... 2-21
- Setting Up the System** .....2-25
  - Creating Job Files ..... 2-25
- Running CIMS Data Collectors** .....2-92
  - Data Processing Frequency ..... 2-92
  - Required Folder Permissions for Data Processing ..... 2-92
  - Running CIMS Job Runner ..... 2-93

# System Specifications

The following are the system specifications for *running* CIMS Data Collectors. Note that you can use CIMS Data Collectors to process usage metering data collected from *any* application or operating system.

- Microsoft Windows 2000 Server or Windows Server 2003 with the latest service pack. (If you are using Windows NT, contact CIMS Lab for assistance.)
- Microsoft .NET Framework 1.1. The CIMS Server setup program (which installs CIMS Data Collectors) includes an option to install the *.NET Framework*. You can install the .NET Framework when you install the data collectors or you can download .NET free-of-charge from the Microsoft Web site, <http://v4.windowsupdate.microsoft.com>.
- Microsoft Windows Script Host (WSH) 5.1 or 5.6 (preferred). You can download both versions free-of-charge at <http://www.microsoft.com>. WSH 5.1 is standard with Windows 2000 Server and will be upgraded to 5.6 if you upgrade to Microsoft Internet Explorer 6 Service Pack 1. WSH 5.6 is standard with Windows Server 2003.

## Installing CIMS Data Collectors

The CIMS Server setup program includes CIMS Data Collectors. When you install CIMS Server, you can choose to install all or certain CIMS Data Collectors (refer to the *CIMS Server Administrator's Guide* for the installation procedures).

You must install all CIMS Data Collectors that you want to use on the central server with the CIMS Server application. Installation on a central server enables you to use CIMS Processing Engine to process the CIMS Server Resource (CSR) files generated by the collectors. CIMS Processing Engine is also included in the CIMS Server installation.

---

**Note** • CIMS Processing Engine is also used to process CIMS Server Resource Plus (CSR+) files, which are created by CIMS Mainframe Data Collector and Chargeback System 12.0 and later.

---

In addition to installation on the central server, you can also install individual collectors on other computers in the following situations.

- You want to use the CIMS Windows Process collector or CIMS Windows Print collector, which produce log files containing operating system and print data, on another computer.

CIMS Lab provides simple setup programs for installing the CIMS Windows Process and Print collectors on other computers. These setup programs install the executable and administrative programs and conversion script for the collector and CIMS Aggregation Engine. For installation and setup procedures for these collectors, see *Chapter 3, Operating System Data Collectors* and *Chapter 9, Printer Data Collectors*.



- You want to convert usage data to a CSR file on the computer that generated the usage data. In most cases, this conversion is performed on the central server.

To perform this conversion, you need to install the conversion file for the collector and CIMS Aggregation Engine on the computer. Contact CIMS Lab for assistance.

## CIMS Data Collectors Architecture

The following is an overview of the components that comprise the CIMS Data Collectors architecture. These components are described in detail in the following sections.

The components are grouped by folder in C:\Program Files\CIMSLab (if you installed CIMS Data Collectors in the default location). Each folder contains the files needed to process usage data. It might be helpful to refer to the folders as you read the following sections.

---

**Important!** • With the exception of the sample job files in the JobFiles folder, the files provided with CIMS Data Collectors usually do not require modification. However, if you modify *any* file that is provided in the CIMSLab folder, it is very important that you rename the file. Otherwise, the file will be overwritten when you upgrade to a new version of CIMS Data Collectors.

---

## CIMS Job Runner Program

CIMS Job Runner is a console application that runs the data collection process. CIMS Job Runner executes jobs that are defined in a job file in the JobFiles folder. Each job can run one or more data collectors.

CIMS Job Runner (CIMSJobRunner.exe) is in C:\Program Files\CIMSLab\Process Engine (if you installed CIMS Data Collectors in the default location).

You can run CIMS Job Runner directly from the command prompt or you can use Windows Task Scheduler to schedule the program to run automatically (see [Running CIMS Job Runner](#) on page 2-93).

## Specifying Log Dates for Collection

CIMS Data Collectors use the LogDate parameter to specify the date for the data that you want to collect. Valid values for the LogDate parameter are:

- PREDAY (Collects data produced on the previous day. This is the default. If you do not provide a LogDate parameter, this value is used.)
- RNDATE (Collects data produced on the current day.)
- PREWEK (Collects data produced in the previous week [Sun–Sat].)
- PREMON (Collects data produced in the previous month.)
- CURWEK (Collect data produced in the current week [Sun–Sat].)
- CURMON (Collects data produced in the current month.)
- date in yyyyymmdd format (Collects data produced on a specified date.)

- date in yyyypp format (Collects data produced in a specified period as defined by the CIMSCalendar table. This is used by the Transactions collector only.)
- date range in yyyymmdd yyyymmdd format (Collects data produced in a specified date range.)

Depending on the collector, these values can be passed by default, at the command line when running CIMS Job Runner, or through the job file as described in the following sections.

#### Passing the Default LogDate Parameter PREDAY

If you are running collectors that process usage metering files on a daily basis, you do not need to provide the LogDate parameter. By default, CIMS Job Runner will collect files created on the previous day. This is the equivalent of using the LogDate parameter PREDAY.

#### Passing the LogDate Parameter from the Command Line

If you need to use a LogDate parameter other than PREDAY, for example you want to process and backload old log files, include the LogDate parameter at the command line when you run CIMS Job Runner (see [Running CIMS Job Runner](#) on page 2-93).

When you enter a LogDate parameter that includes a date range, such as CURMON, CIMS Job Runner runs the data collection process for each day in the range. If log file generation and e-mail messaging is enabled in the job file, a separate log file and e-mail message is generated for each day.

#### Passing the LogDate Parameter from the Job File

The LogDate parameter should be included in the job file only in the following situations:

- **You are running a snapshot collector.** The snapshot data collectors (DBSpace, CIMS Windows Disk, and Exchange Server Mailbox) collect data that is current as of the date and time that the collectors are run by CIMS Job Runner. However, the start and end date that appears in the output CSR file records and the date that appears in the initial CSR file name will reflect the LogDate parameter value. For example, if you use the LogDate parameter PREDAY, the previous day's date is used.

If you want the actual date that the data was collected to appear in the CSR file, you need to use the keyword RNDATE as the LogDate parameter. When RNDATE is specified in the job file, you must ensure that the command line does not include a LogDate parameter or that RNDATE is provided at the command line. Log date values provided in the command line will override values in the job file.

- **You are running the Transactions collector.** The Transactions collector uses the LogDate parameters CURMON, PREMON, or the date/period in yyyypp format only. The yyyypp format is specific to the Transactions collector and cannot be passed from the command line. In addition, CURMON and PREMON cannot be passed from the command line for the Transactions collector.

For more information about the Transactions collector, see [Chapter 10, Transactions Collector](#).

## Job Files (JobFiles Folder)

A job file is an XML file that defines the data collection process. The job file definitions include the applications that you want to collect usage data for and the location of the applications. The job file also defines the conversion file to be used to convert the data and the other CIMS components required to process the data and load it into a CIMS Server database. (For a description of how the database is determined, see [page 2-49](#) for a description of the `dataSourceId` attribute in the job file.)

CIMS Lab provides a sample job file, `SampleNightly.xml`, that you can modify for your organization. If you modify this file, you need to rename it (for example, `Nightly.xml`). Otherwise, the file will be overwritten when you install a new version of CIMS Data Collectors. If you installed CIMS Data Collectors in the default location, the `SampleNightly.xml` file is in `C:\Program Files\CIMSLab\JobFiles`.

The `SampleNightly.xml` job file is intended to be run on a nightly basis to run one or multiple data collectors. However, you can schedule CIMS Job Runner to run job files on any schedule. CIMS Lab also provides a sample job file for monthly data collection and processing, `SampleMonthly.xml`. You must also rename this file before you modify it.

For a description of the job file structure, see [Creating Job Files](#) on page 2-25.

## Job File XML Schema

---

**Important!** • Do not modify this file.

---

The job files use an XML schema, `CIMSJob.xsd`. This schema defines and validates the structure of the job file(s). The definitions in the schema include the following:

- The elements that can appear in the job file.
- The attributes that can appear in the job file.
- Which elements are child elements.
- The number and order of child elements.
- Whether an element is empty or can include text.
- Data types for elements and attributes.
- Default and fixed values for elements and attributes.

The file `CIMSJobs.xsd` is in `C:\Program Files\CIMSLab\Process Engine\JobLibrary` (if you installed CIMS Data Collectors in the default location).

### Collection Files (Collectors Folder)

Collection files are used to collect and convert usage data produced by an application. The `Collectors` folder contains a subfolder for each CIMS Data Collector. Depending on the collector, each subfolder contains one or more of the files described in the following sections.

---

**Important!** • If you modify any file/script in the `Collectors` folder, you need to rename the file. Otherwise, the file will be overwritten when you upgrade to a new version of CIMS Data Collectors.

---

#### Conversion Script

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

- Calls CIMS Aggregation Engine (if applicable). CIMS Aggregation Engine (`CIMSAggregation.dll`) is a Component Object Model (**COM**) object 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.

For more information about the CIMS Aggregation Engine, see [Appendix A, CIMS Aggregation Engine API](#).

- 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.

- Places the output CSR file in a feed subfolder to be processed by the Scan program.

## 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 <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	<p>Use this parameter to specify the source that contains the usage metering data to be collected. Usually, this is the name of the server that contains the data. (For example, if you are collecting data from a server named Server1, use Server1 as the feed name). However, depending on the collector, it might be another source. For example, for the CIMS Windows Disk collector, the Feed parameter should include the drive or folder that contains the data (for example, Server1-C). The Feed parameter requirement for each collector is provided in the following chapters.</p> <p>A subfolder with the same name as the feed is automatically created in the <a href="#">process definition folder</a> (see the OutputFolder parameter). This subfolder is used to store the initial CSR file that is created from a usage metering file (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p><b>Note:</b> Although this subfolder is created in the Transaction process definition folder, it is not used. CSR files are placed directly within the process definition folder.</p> <p>This parameter is included as an identifier in the CSR file with the exception of resource files created from transactions or the Universal collector. The identifier name is Feed and the identifier value is the server name.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program (see <a href="#">page 2-9</a>).</p> <p>This parameter is defined by the Process id for the collector. For example, Process id="MSIIS-Web" specifies that the output folder is the MSIIS-Web process definition folder. For more information about the process definition folder, see <a href="#">page 2-12</a>.</p>

**Table 2-1 • Conversion Script Parameters**

In addition to the parameters in the preceding table, other parameters are required or optional depending on the specific collector. Parameters that are collector-specific are described in the following chapters.

### **Executable and Other Programs**

Depending on the collector, the collector subfolder might contain any of the following files: an installation program, an executable program, and/or a graphical user interface (GUI) program to configure the collector.

## **Job Log Files (LogFiles Folder)**

A log file is created for each job that you run. This log file provides processing results for each step defined in the job file. If a warning or failure occurs during processing, the file indicates at which point the warning/failure occurred.

---

**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 file contains a syntax error) a log file is not generated. To ensure that the job runs correctly and that a log file is generated, you can run the job file from the command line (see [page 2-94](#)).

---

Within the LogFiles folder, individual log files are automatically stored in a subfolder with the same name as the job that generated the log. For example, if the job ID in the job file is "Nightly", the log files are stored in the Nightly subfolder. For more information about defining a job, see [page 2-48](#).

The log file name contains the date and time that the file was created.

### **Defining the Log File Output Type**

You can produce log data in a text file, an XML file, or both. To define the type of file that you want to use for log data, you need to set the attributes `joblogWriteToTextFile="true"` and/or `joblogWriteToXMLFile="true"` in the job file (see [page 2-50](#)).

### **Defining the Log File Content**

You can choose to exclude some data from the log file to reduce the log file size. To specify data that you do not want to appear in the log file, you need to set the attributes `joblogShowStepParameters="false"` and/or `joblogShowStepOutput="false"` in the job file (see [page 2-49](#)).

### **Sending Log Files Via E-Mail**

You can choose to have output log files sent via e-mail to a recipient or recipients. To send log files via e-mail, you need to set the appropriate SMTP definitions in the job file (see [page 2-51](#)).

## Job Log Return Codes

The log file provides the following return codes for each step in the job file. These codes specify whether the step completed successfully, completed with warnings, or failed.

- 0 Execution ended with no errors or warnings.
- 4 or 8 Execution ended with warning messages.
- 16 Execution ended with errors—processing stopped.

## CIMS Processing Programs (Process Engine\JobLibrary Folder)

The components in the JobLibrary folder are used by the following programs. These programs are defined in the job file. The following programs are required:

- Scan (if you are collecting from multiple servers).
- CIMSAct, CIMSSort, and CIMSBill or SingleProcessStep.
- DBLoad.

All other programs are optional.

### Scan

The Scan program performs the following tasks:

- Verifies that the feed subfolder or subfolders in a process definition folder contain a CSR file that matches the LogDate parameter. If a matching file is not found, a warning or error occurs depending on the job file definition (see [page 2-62](#)).
- Concatenates the CSR files produced by data collectors of the same type from multiple servers into one file.
- Outputs a CSR file (whether from one server or a concatenated file from multiple servers) to the collector's process definition folder. The default file name for the CSR file is CurrentCSR.txt.

---

**Important!** • If you are collecting from only one server, the use of the Scan program is optional. However, if you do not use this program, you need to move the CSR file contained in the feed subfolder to the collector's process definition folder.

---

For the parameters used by the Scan program, see [page 2-62](#).

The Scan program has an option called Smart Scan. For more information about this option, see [Example of Reprocessing With Smart Scan Enabled](#) on page 2-42.

### CIMSAct

The CIMSAct program performs account code conversion, shift determination, date selection, and identifier extraction on the usage data, and produces the CIMSAct Detail file containing records that are properly formatted for input into CIMSBill.

For the parameters used by the CIMSAct program, see [page 2-65](#).

### **CIMSSort**

The CIMSSort program sorts the CIMSActt Detail file and produces a version of the file that is ready to be processed by CIMSBill.

For the parameters used by the CIMSSort program, see [page 2-70](#).

### **CIMSBill**

The CIMSBill program processes the sorted CIMSActt Detail file from CIMSSort and performs shift processing, CPU normalization, and include/exclude processing and creates the CIMSBill Detail and CIMS Summary files. These files contain the billing information used to generate invoices and reports.

---

**Note** • Although you can perform proration using CIMSBill, CIMS Lab recommends that you use the CIMSPrat program. CIMSPrat provides more options for proration and enables you to prorate resources multiple times. CIMS Server will continue to support proration using CIMSPrat or CIMSBill.

---

For the parameters used by the CIMSBill program, see [page 2-71](#).

### **SingleProcessStep**

The SingleProcessStep program calls CIMSActt, CIMSSort, and CIMSBill using default parameters.

### **DBLoad**

The DBLoad program loads the output files from CIMSActt and CIMSBill into the CIMS Server database. (For a description of how the database is determined, see [page 2-49](#) for a description of the dataSourceId attribute in the job file.)

For the parameters used by the DBLoad program, see [page 2-76](#).

### **CIMSPrat**

The CIMSPrat program processes the resources in CSR and CSR+ file records and creates a new file with prorated resources. Proration enables you to distribute resources and costs for a specified rate code or codes or all rate codes across multiple accounts at a specified percentage.

For the parameters used by the CIMSPrat program, see [page 2-64](#).

To run CIMSPrat, you need a proration table and an XML file that contains the parameters used by CIMSPrat. These files are described in [Setting Up Proration Files \(Optional\)](#) on page 2-15.



## **WaitFile**

The WaitFile program directs CIMS Job Runner to wait for one or more files before continuing processing. For example, if you are collecting mainframe files produced by CIMS Mainframe Data Collector and Chargeback System, you can specify the length of time to wait for the files.

For the parameters used by the WaitFile program, see [page 2-78](#).

## **FileTransfer**

The FileTransfer program transfers one or more files from one computer to another. For example, you can use this program to pull files from mainframe or UNIX systems to the central CIMS Data Collectors server.

For the parameters used by the FileTransfer program, see [page 2-80](#).

## **Cleanup**

The Cleanup program deletes files with file names containing the date in yyyyymmdd format in the collector's process definition folder or any other folder that you specify (for example, the folder that contains an application's log files). You can use the Cleanup program to delete files after a specified number of days from the file's creation or to delete files that were created before a specified date.

For the parameters used by the Cleanup program, see [page 2-85](#).

### Process Definitions (Processes Folder)

A process definition is a subfolder within the Processes folder. (The Processes folder is shipped as Sample Processes, see [About the Processes Folder](#) on page 2-12).

Process definition folders contain the files required to process usage data from a particular source such as a database, operating system, or application. You can modify and maintain these files using CIMS Server Administrator (refer to the *CIMS Server Administrator's Guide*). Process definition folders are also used to store the CSR files that are generated from the usage data.

A separate process definition folder is required for each application that you collect data from. If a process definition folder does not exist for the collector, CIMS Job Runner can create a folder using the process ID defined in the job file as the folder name (see [page 2-29](#)).

#### About the Processes Folder

The folder Sample Processes is shipped with CIMS Data Collectors. When CIMS Data Collectors are installed for the first time, this folder should be renamed/and or moved before any of its subfolders or files are modified. CIMS Lab recommends that the folder be renamed Processes and moved to a location where the folder will be backed up. However, the folder can be given any name and moved to any location. This folder is referred to as Processes in this guide.

---

**Important!** • The path to the Processes folder must be defined in the processing definition path setting in CIMS Server Administrator. The path is stored in the CIMSCfgOptions table and is used as the default path for the process definition folders. To set the Processes folder path, refer to the *CIMS Server Administrator's Guide*.

---

Each time that you upgrade to a new release of CIMS Data Collectors, a new Sample Processes folder is installed. You can then copy or move any new process definition folders that you want from the Sample Processes folder to the Processes folder. Each process definition folder contains the files and subfolders described in the following sections.

## Feed Subfolder

A feed subfolder is automatically created in the process definition folder for each server that you entered as a `Feed` parameter in the job file. If you left the `Feed` parameter blank or did not include the parameter, the feed subfolder is named `Server1`.

---

**Note** • For the CIMS Windows Disk collector, a value is required for the `Feed` parameter (i.e., you cannot leave this parameter blank). For more information about this collector, see [Chapter 7](#).

---

Each feed subfolder is used to store CSR files from the feed of the same name. The CSR file name contains a date in `yyyymmdd` format. Note that although the feed subfolder is created in the `Transactions` process definition folder, it is not used. CSR files created by the `Transactions` collector are placed directly in the process definition folder. For more information, see [Chapter 10](#).

The `Scan` program processes and concatenates the CSR files in the feed subfolders as described in [Scan](#) on page 2-9. The resulting output file is placed directly in the process definition folder.

---

**Important!** • To prevent data processing errors, the process definition folder should not contain subfolders other than feed folders and feed folders should not contain files other than CSR files.

---

## XML File

The CIMS Windows Event Log and CIMS Windows Disk collectors include an XML file (`CIMScollectorname.xml`) that provides parameters used by these collectors. For more information, see [Chapter 7](#) and [Chapter 9](#).

## Additional Processing Files

Each process definition folder contains additional processing files that are used internally by CIMS Data Collectors.

### Scripts (Scripts Folder)

The CIMS Data Collectors architecture includes the following scripts that support processing tasks.

#### **CIMSUtils.wsc**

The Windows Script Component file `CIMSUtils.wsc` provides useful utilities and tools including methods for getting the Processes and CIMSLab folders and for building the Open Database Connectivity (*ODBC*) connection string.

---

**Note** • By default, the script component files `CIMSUtils.wsc` and `Shell.wsc` are registered on your computer as COM objects at installation. However, if you move these files to another computer, you need to register the files again.

---

#### **Shell.wsc**

The `Shell.wsc` file is used to capture messages passed from one script to another. This file is used for legacy purposes only.

#### **CIMSLIB.wsf**

`CIMSLIB.wsf` is a library of Windows Scripting Functions that can be used in the script files. This file is used as an include file.

## Setting Up Proration Files (Optional)

---

**Note** • Proration is an optional feature. Skip this section if you do not want to prorate resources in a CSR or CSR+ file.

---

Proration is taking the overall or individual resources used by an account and distributing those resources and the cost of the resources across multiple accounts at a specified percentage.

A common use of proration is the equitable allocation of overhead costs across the user community. An application that is normally assigned to the overhead category tends to be one that does not produce metering data to the end user or account level, or the processing of such data is considered too expensive to be practical at the end user or account level. For example, an application that controls print in a centralized environment.

Although you can perform proration using CIMSBill, CIMS Lab recommends that you use the CIMSPrat program. CIMSPrat provides more options for proration and enables you to prorate a CSR or CSR+ file multiple times.

CIMSPrat processes a CSR or CSR+ file for an application and produces a new file with prorated resource units. To prorate resources using CIMSPrat, you need to create the following files:

- **A proration table.** This table contains comma-delimited records that define the identifier values and rate codes used in the proration process. See *Creating a Proration Table* on page 2-16.
- **An XML file that contains the parameters used by CIMSPrat.** This file provides the input and output file parameters and processing parameters required to produce prorated CSR or CSR+ files. See *Creating the CIMSPrat Parameters File* on page 2-17.

The proration table must be referenced in the CIMSPrat parameters file. For an example of how these files work together to produce the prorated CSR or CSR+ file, see *Proration Example* on page 2-21.

As with CIMSAct, CIMSBill, and the other CIMS programs, CIMSPrat is run as a step in a job file as shown in the example on page [page 2-33](#).

## Creating a Proration Table

The CIMS Server installation includes a sample proration table, `Prorate.txt`, in the `Prorate` process definition folder. You can modify the `Prorate.txt` table or you can create a new proration table. If you modify the `Prorate.txt` file, you need to rename and/or move the file so that it is not overwritten when you upgrade to a new version of CIMS Data Collectors.

The proration table must contain records with the following comma-delimited fields:

```
input identifier value,output identifier value,percentage,rate code
```

The value in the input identifier value field is matched against an identifier value in the input CSR or CSR+ file. The identifier name used to match the identifier value is defined by the `IdentifierName` attribute in the `CIMSPrat` parameters file (see [page 2-18](#)).

The output identifier value is the value that appears in the prorated CSR or CSR+ file records.

The percentage is the percentage of resource units that you want to prorate.

The rate code can be a specific rate code or all rate codes. If you specify a specific rate code, just that rate code in the selected records are prorated. If you specify `All` or leave this field blank, the resources for all rate codes in the selected record are prorated.

For an example of a proration table used in the proration process, see [Proration Example](#) on page 2-21.

## Creating the CIMSPrat Parameters File

The CIMS Server installation includes a default CIMSPrat parameters file, `CIMSPrat.xml`, in the `Prorate` process definition folder. You need to edit the following parameter attributes in this file for your organization and then rename and/or move the file so that it is not overwritten when you upgrade to a new version of CIMS Data Collectors.

For an example of a modified CIMSPrat parameters file used in the proration process, see *Proration Example* on page 2-21.

Attribute	Required or Optional	Description
InputFile and OutputFile	Required	These parameters should be set to the same value: the path for the process definition folder that contains the CSR or CSR+ file that you want to prorate. The prorated CSR or CSR+ file will also be placed in this folder.
ProrateFile	Required	The full path for the proration table. The proration table can be in any folder.
PrintFile	Required	The full path for the output CIMSPrat processing report. This report provides detailed information related to the CIMSPrat run including the parameters and proration table used and the process results.  The CIMSPrat report can be in any folder.
ExceptionFile	Required	The full path for the output exception file. This file contains records that do not include an identifier name that matches the <code>IdentifierName</code> attribute value. The exception file can be in any folder.  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 proration table, and then reprocess the exception file.  To enable the creation of an exception file, you must have the <code>ExceptionProcessing</code> attribute set to "TRUE". If you have <code>ExceptionProcessing</code> set to "False", unmatched records are included in the prorated CSR or CSR+ file.

Table 2-2 • CIMSPrat.xml Parameters

## ■ Installing CIMS Data Collectors and Setting Up the System

### Setting Up Proration Files (Optional)

Attribute	Required or Optional	Description
Audit	Optional	<p>Specifies whether the following audit information is included in the prorated records. This information is provided as additional identifiers in the record (see the example prorated records on <a href="#">page 2-23</a>).</p> <ul style="list-style-type: none"> <li>■ The input identifier name prefixed by <code>Orig_</code> and the original identifier value. (This is not applicable if the <code>DiscardIdentifier</code> is set to "TRUE" and/or <code>NewIdentifier</code> attribute is set to a new value).</li> <li>■ An additional rate code, <code>ProratePct</code>, that provides the proration percentage value.</li> <li>■ The input rate code prefixed by <code>Orig_</code> and the original resource value.</li> </ul> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "TRUE" (audit is enabled)</li> <li>■ "FALSE" (audit is not enabled)</li> </ul> <p>The default is "TRUE".</p>
IdentifierName	Required	<p>The name of the identifier field that you want to use to select CSR or CSR+ records for proration.</p> <p>Records that contain this identifier name are matched to the entries in the proration table. If the identifier value in the record matches an input identifier value in the proration table, the record is prorated as specified in the table.</p>
AllowNon100Totals	Optional	<p>Specifies whether the total prorate percentages for an identifier name must equal 100 percent. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "TRUE" (percentages are not required to equal 100 percent)</li> <li>■ "FALSE" (percentages must equal 100 percent)</li> </ul> <p>The default is "TRUE".</p>

**Table 2-2 • CIMSPrat.xml Parameters (Continued)**



Attribute	Required or Optional	Description
IdentifierStart	Optional	<p>The position in the identifier value field of the CSR or CSR+ record that you want to begin comparing to the input identifier value field in the proration table.</p> <p>The default is 1.</p>
IdentifierLength	Optional	<p>The number of characters in the identifier value field of the CSR or CSR+ record that you want to compare to the input identifier value field in the proration table.</p> <p>This value begins at the position specified by the IdentifierStart attribute.</p> <p>The default is 8.</p>
MaximumRecords	Option	<p>The maximum number of records that you want to process in the CSR or CSR+ files.</p> <p>The default is all records in the file.</p>
PrintLines	Optional	<p>The number of lines per page in the CIMSPrat processing report.</p> <p>The default is 60.</p>
ExceptionProcessing	Optional	<p>Specifies whether the exception file should be created. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "TRUE" (the exception file is created)</li> <li>■ "FALSE" (the exception file is not created)</li> </ul> <p>The default is "FALSE".</p>
NewIdentifier	Optional	<p>Specifies that a new identifier name will appear for the output identifier value in the prorated records.</p> <p>The default is to use the original identifier name.</p> <p>For an example of the use of this attribute, see <i>Proration Example</i> on page 2-21.</p>

Table 2-2 • CIMSPrat.xml Parameters (Continued)

## ■ Installing CIMS Data Collectors and Setting Up the System

### Setting Up Proration Files (Optional)

Attribute	Required or Optional	Description
DiscardIdentifier	Optional	<p>Specifies that the original identifier name and value will not appear in the prorated records.</p> <p>Valid values are:</p> <ul style="list-style-type: none"><li>■ "TRUE" (the original identifier name and value are discarded)</li><li>■ "FALSE" (the original identifier name and value are retained)</li></ul> <p>The default is "FALSE".</p> <p>This attribute is useful in situations where the original identifier value is not intended for public view (for example, a social security number).</p> <p>This attribute is intended to be used with the <code>NewIdentifier</code> attribute. If you do not include the <code>NewIdentifier</code> attribute, neither the input nor output identifier names and values will appear in the prorated records.</p> <p>For an example of the use of this attribute, see <a href="#">Proration Example</a> on page 2-21.</p>
Test	Optional	<p>A numeric value used as a test flag. Used by CIMS Lab technical support only.</p>
<p><b>The following are catchall parameters. These parameters are used for records with identifier values that do not have a matching entry in the proration table. You can specify one set of catchall parameters or multiple sets.</b></p> <p><b>To use these parameters, you must remove the comments and make sure that the parameters are in the correct format. For an example of the use of these parameters, see <a href="#">page 2-22</a>.</b></p>		
CatchallIdentifier	Optional	<p>Specifies the identifier value to be used for records with identifier values that do not have a match in the proration table.</p> <p>If you want to use catchall processing, you must include this parameter, regardless of whether you specify a value or accept the default (<code>Catchall</code>).</p> <p>Once you uncommented this parameter, you can leave the remaining catchall parameters commented, and the defaults will be used.</p>

Table 2-2 • CIMSPrat.xml Parameters (Continued)

Attribute	Required or Optional	Description
CatchallPercent	Optional	Specifies the proration percentage to be used for records with identifier values that do not have a match in the proration table.  The default is 100.
CatchallRate	Optional	Specifies the rate code(s) to be used for records with identifier values that do not have a match in the proration table.  The default is all rate codes.

Table 2-2 • CIMSPrat.xml Parameters (Continued)

## Proration Example

Your organization uses software program, ABCPrint, that produces usage metering data by printer. Using CIMS Universal Data Collector described in [Chapter 14, CIMS Universal Data Collector](#), you have converted the log file produced by this software into a CSR file.

The records in the CSR file contain one identifier PrinterName, and two rate codes, SUBBYTE and PRNTBYTE, as shown in the following example:

```
ABCPrint,20050916,20050916,13:01:50,13:01:50,,1,PrinterName,"LaserJ",2,SUBBYTE,5107,PRNTBYTE,5107
ABCPrint,20050916,20050916,16:15:03,16:15:03,,1,PrinterName,"OptraL",2,SUBBYTE,1913,PRNTBYTE,1913
ABCPrint,20050916,20050916,17:13:33,17:13:33,,1,PrinterName,"PhaserDX",2,SUBBYTE,2525,PRNTBYTE,2525
```

You want to redistribute the resources for the rate codes SUBBYTE and PRNTBYTE that are currently assigned to printers LaserJ, OptraL, and PhaserDX to individual users or accounts. To do this, you need to create a proration table and CIMSPrat parameters file as shown in the following examples.

### Proration Table Example

In the following example table, all resources for printer LaserJ are prorated to three users for a total of 100 percent. For printer OptraL, only the resources for rate code PRNTBYTE are prorated and the resources are prorated to four users. The total proration for both LaserJ and OptraL is 100 percent; however, a proration total of 100 percent is not required (see the AllowNon100Totals attribute on [page 2-18](#)).

For example purposes, the prorate table does not contain entries for the printer PhaserDX. This record will be processed using catchall parameters in the CIMSPrat parameters file (see [page 2-22](#)).

```
LaserJ,MikeR,25,A11
LaserJ,Robert,25,A11
LaserJ,Joan,50,A11
OptraL,MikeL,25,PRNTBYTE
OptraL,Bill,25,PRNTBYTE
OptraL,Mark,25,PRNTBYTE
OptraL,Tom,25,PRNTBYTE
```

## CIMSPrat Parameters File Example

Assume that the CIMSPrat parameters file, CIMSPrat.xml, has been modified as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<CIMSPrat version="1.0">
  <!-- For file names, full path name may be used. -->
  <!-- InputFile - file read in and processed. -->
  <!-- OutputFile - file with matching prorated records -->
  <!-- ProrateFile - file with proration control cards -->
  <!-- PrintFile - Output report file -->
  <!-- ExceptionFile - File where unmatched records are sent -->
  <Parameter InputFile="C:\Program Files\CIMSLab\Processes\ABCPrint\CurrentCSR.TXT"
    OutputFile="C:\Program Files\CIMSLab\Processes\ABCPrint\ProRatedCurrentCSR.TXT"
    ProrateFile="C:\Program Files\CIMSLab\Processes\ABCPrint\Prorate.txt"
    PrintFile="C:\Program Files\CIMSLab\ABCPrint\Processes\PRATPRNT.TXT"
    ExceptionFile="C:\Program Files\CIMSLab\Processes\ABCPrint\PRATEXCP.TXT" />

  <!-- Audit=TRUE/FALSE - Indicates whether or not to write original fields as audit trail. Default
  is TRUE. -->
  <Parameter Audit="TRUE" />

  <!-- IdentifierName= - Name of identifier field to search. -->
  <Parameter IdentifierName="PrinterName" />

  <!-- AllowNon100Totals=TRUE/FALSE - Indicates whether or not total proration percentages must equal
  100% -->
  <Parameter AllowNon100Totals="TRUE" />

  <!-- IdentifierStart= - First position in field to check. Default 1. -->
  <Parameter IdentifierStart="1" />

  <!-- IdentifierLength= - Number of characters to compare. Default to the entire field. -->
  <Parameter IdentifierLength="6" />

  <!-- MaximumRecords= - Maximum number of records to process. Default is process entire file. -->
  <Parameter MaximumRecords="" />

  <!-- PrintLines= - Lines per page. Default is 60. -->
  <Parameter PrintLines="" />

  <!-- ExceptionProcessing=TRUE/FALSE - Indicates whether to suppress exception file. Default is
  FALSE. -->
  <Parameter ExceptionProcessing="TRUE" />

  <!-- NewIdentifier= - New identifier field name to assign to updated field. If not specified,
  original name will be used. -->
  <Parameter NewIdentifier="User" />

  <!-- DiscardIdentifier=TRUE/FALSE - Indicates whether to drop identifier field used for search.
  Default is FALSE. -->
  <Parameter DiscardIdentifier="TRUE" />

  <!-- Test= - Numeric value used for a test flag. Only to be used by CIMS Lab technical support. -->
  <Parameter Test="0" />

  <!-- CatchallIdentifier= - Identifier to be used if there is no match for the identifier field in
  the proration table. Default if left null is CATCHALL. -->
  <!-- CatchallPercent= - Percentage to be used. Default is 100. -->
  <!-- CatchallRate= - Rate code to be prorated. Default is all rate codes. -->
  <!-- There may be more than one set of catchall parameters specified. -->
  <!-- If no catchall parameters are specified, catchall processing will not be used. -->
  <Parameter CatchallIdentifier="Unassigned_Mktg" />
  <Parameter CatchallIdentifier="Unassigned_Sales" />
  <!--Parameter CatchallIdentifier="CATCH30" -->
```

```

<!--Parameter CatchallIdentifier="CATCH40" -->
<Parameter CatchallPercent="50" />
<Parameter CatchallPercent="50" />
<!--Parameter CatchallPercent="30" -->
<!--Parameter CatchallPercent="40" -->
<!--Parameter CatchallRate="" -->
<!--Parameter CatchallRate="" -->
<!--Parameter CatchallRate="" -->
<!--Parameter CatchallRate="" -->
</CIMSPrat>

```

## Prorated Records Example

Using the preceding proration table and parameters file, CIMSPrat would produce a prorated CSR file with the following records. (The records are numbered for example purposes only). For more information about these records, see [page 2-24](#).

```

1 ABCPrint,20050916,20050916,13:01:50,13:01:50,,4,User,MikeR,ProratePct,25,Orig_SUBBYTE,5107,
  Orig_PRNTBYTE,5107,2,SUBBYTE,1276.75,PRNTBYTE,1276.75
2 ABCPrint,20050916,20050916,13:01:50,13:01:50,,4,User,Robert,ProratePct,25,Orig_SUBBYTE,5107,
  Orig_PRNTBYTE,5107,2,SUBBYTE,1276.75,PRNTBYTE,1276.75
3 ABCPrint,20050916,20050916,13:01:50,13:01:50,,4,User,Joan,ProratePct,50,Orig_SUBBYTE,5107,
  Orig_PRNTBYTE,5107,2,SUBBYTE,2553.5,PRNTBYTE,2553.5
4 ABCPrint,20050916,20050916,16:15:03,16:15:03,,3,User,MikeL,ProratePct,25,Orig_PRNTBYTE,1913,1,
  PRNTBYTE,478.25
5 ABCPrint,20050916,20050916,16:15:03,16:15:03,,3,User,Bill,ProratePct,25,Orig_PRNTBYTE,1913,1,
  PRNTBYTE,478.25
6 ABCPrint,20050916,20050916,16:15:03,16:15:03,,3,User,Mark,ProratePct,25,Orig_PRNTBYTE,1913,1,
  PRNTBYTE,478.25
7 ABCPrint,20050916,20050916,16:15:03,16:15:03,,3,User,Tom,ProratePct,25,Orig_PRNTBYTE,1913,1,
  PRNTBYTE,478.25
8 ABCPrint,20050916,20050916,16:15:03,16:15:03,,1,PrinterName,OptraL,PrinterName,OptraL,PrinterName,
  OptraL,PrinterName,OptraL,1,SUBBYTE,1913
9 ABCPrint,20050916,20050916,17:13:33,17:13:33,,4,User,Unassigned_Mktg,ProratePct,50,Orig_SUBBYTE,
  2525,Orig_PRNTBYTE,2525,2,SUBBYTE,1262.5,PRNTBYTE,1262.5
10 ABCPrint,20050916,20050916,17:13:33,17:13:33,,4,User,Unassigned_Sales,ProratePct,50,Orig_SUBBYTE,
  2525,Orig_PRNTBYTE,2525,2,SUBBYTE,1262.5,PRNTBYTE,1262.5

```

#### **About Records 1–8**

The original identifier name, `PrinterName`, has been removed from the records and replaced with the identifier name `User` as specified by the `NewIdentifier` and `DiscardIdentifier` attributes in the `CIMSPrat.xml` file. The new identifier name is followed by the output identifier values defined in the proration table.

Because the `Audit` attribute in the `CIMSPrat.xml` file is set to "TRUE", three identifier have been added to the records: `ProratePct`, `Orig_SUBBYTE`, and `Orig_PRNTBYTE`. These identifiers specify the prorate percentage (as defined in the proration table) and the original resource values for the prorated rate codes.

For record 1–3, both rate code `SUBBYTE` and `PRNTBYTE` appear in the record with prorated resources because the matching entry in the proration table contained `All` in the rate code field.

For records 4–7, only the rate code `PRNTBYTE` and its prorated resources appear in the record because the matching entry in the proration table contained this rate code in the rate code field. The rate code and unprorated resources for `SUBBYTE` are provided in record 8. Because the resources in this record were not prorated, the record contains the original identifier name and value.

#### **About Records 9 and 10**

The original identifier name, `PrinterName`, has been removed from the records and replaced with the identifier names `Unassigned_Mktg` and `Unassigned_Sales` as specified by the `CatchallIdentifier` parameter in the `CIMSPrat.xml` file. The percentage and rate codes used for proration were specified by the `CatchallPercent` and `CatchallRate` parameters in the `CIMSPrat.xml` file rather than an entry in the proration table.

## Setting Up the System

**Note** • This section assumes that you have reviewed the CIMS Data Collectors architecture described in *CIMS Data Collectors Architecture* beginning on page 2-3.

### Creating Job Files

**Note** • CIMS Lab provides two sample job files: `SampleNightly.xml` and `SampleMonthly.xml`. If you modify these files, you need to rename them `Nightly.xml` and `Monthly.xml` (or choose other names) so that the files are not overwritten when you upgrade to a new version of CIMS Data Collectors.

A job file is an XML file that specifies which CIMS Data Collectors are run and the data collection process. CIMS Lab provides a two sample job files, `SampleNightly.xml` and `SampleMonthly.xml`, that you can modify for your organization. If you installed CIMS Data Collectors in the default location, these files are in `C:\Program Files\CIMSLab\JobFiles`. For more information about the `SampleNightly.xml` job file, see *Using the SampleNightly.xml Job File* on page 2-27.

Whether you are modifying a sample job file or creating a new file, you need to follow the structure provided in the sample job files (see *Job File Structure* on page 2-46).

### Using CIMS Date Keywords in the Job File

Where applicable in the job file, you can provide a date in `yyyymmdd` format or you can use one of the following CIMS date keywords. The attributes in the log file that can use a date keyword are described in the *Job File Structure* section beginning on page 2-46.

Keyword	Description
RNDATE	The current day.
CURDAY	The current day and the previous day.
CURWEK	The current week (Sun–Sat).
CURMON	The current month.
PREDAY	The previous day.
PREWEK	The previous week (Sun–Sat).
PREMON	The previous month.

### Using Log Dates in the Job File

CIMS Data Collectors use the `LogDate` parameter to specify the date for the data that you want to collect. (For a complete description of the `LogDate` parameter and its valid values, see *Specifying Log Dates for Collection* on page 2-3.) Depending on the collector, the `LogDate` parameter can be passed as a default, at the command line when running CIMS Job Runner, or through the job file. The `LogDate` parameter should be included in the job file only in the following situations:

- **If you are running a snapshot collector.** The snapshot data collectors (DBSpace, CIMS Windows Disk, and Exchange Server Mailbox) collect data that is current as of the date and time that the collectors are run by CIMS Job Runner. However, the start and end date that appears in the output CSR file records and the date that appears in the initial CSR file name will reflect the `LogDate` parameter value. For example, if you use the `LogDate` parameter `PREDAY`, the previous day's date is used.

If you want the actual date that the data was collected to appear in the CSR file, you need to use the keyword `RNDATE` as the `LogDate` parameter. When `RNDATE` is specified in the job file, you must ensure that the command line does not include a `LogDate` parameter or that `RNDATE` is provided at the command line. Log date values provided in the command line will override values in the job file.

- **If you are running the Transactions collector.** The Transactions collector uses the `LogDate` parameters `CURMON`, `PREMON`, or the date/period in `yyyymm` format only. The `yyyymm` format is specific to the Transactions collector and cannot be passed from the command line. In addition, `CURMON` and `PREMON` cannot be passed from the command line for the Transactions collector.

You can enter the `LogDate` parameter at the job, process, or step level depending on whether you want the log date to apply to all steps in a job, all steps in a process, or a specific step. To provide the `LogDate` parameter at the job or process level, you need to use the `Default` element as described in *Default Element (Optional)* on page 2-90. To provide the `LogDate` parameter at the step level, you need to use the `Parameter` element (for an example, see page 10-4).



## Using the SampleNightly.xml Job File

The purpose of the SampleNightly.xml job file is to show how to run a variety of CIMS Data Collectors and to show the different configurations that you might want to implement for the data collection process.

In Table 2-3, the first column shows the data that is collected by the SampleNightly.xml job file and the second column shows the example configuration that is provided for collection of that data and provides a link to the appropriate process in the job file. Note that the configuration examples are not collector specific.

Data Collected	Configuration Example	Description
IIS	Collecting from multiple servers.	The example XML for this process collects IIS log files that are contained on two servers.
Exchange Server	Transferring a file.	<p>The example XML for this process includes steps to transfer Exchange Server log files from their source location on separate servers to a location in which the files can be processed by the Exchange Server collector.</p> <p>For example purposes, the first file transfer is an FTP transfer and the second transfer is a Windows transfer.</p> <p>Transferring a file is useful when the source computer is behind a firewall or is otherwise not accessible from the central CIMS Data Collectors server.</p>
Mainframe	Waiting for a file.	The example XML for this process includes a step that instructs CIMS Job Runner to wait for the CSR+ file from CIMS Mainframe before continuing processing.
UNIX	Prorating resources.	The example XML for this process includes a step that prorates the resources in the CSR file and creates a new file named ProratedCSR.txt. The prorated file is used as input to CIMSActt.

Table 2-3 • SampleNightly.xml Job File Contents

Data Collected	Configuration Example	Description
<b>SQL Server</b>	Using separate CIMSActt, CIMSSort, and CIMSBill steps.	<p>The example XML for this process includes separate steps for the CIMSActt, CIMSSort, and CIMSBill programs rather than a single step for the SingleProcessStep program.</p> <p>Note that the parameters for each program are included in comments for example purposes. You do not need to include parameters unless you want to change the default values.</p>
<b>Disk Storage Event Log</b>	Creating an external XML from within the job file.	<p>Both the CIMS Windows Disk collector and the CIMS Windows Event Log collector require an external XML file. The external file is built within the job file as shown in the CIMSWinDisk and CIMSWinEventLog processes.</p>
<b>Database Size</b>	Loading specific files into the database rather than all files.	<p>The example XML for this process loads only the CIMS Ident and CIMSBill Detail files into the database. The CIMS Summary file is not loaded.</p> <p>The XML also shows how to optionally specify a name for the CIMS Ident and CIMSBill Detail files other than the default name. In this example, the files are named MyIdent.txt and MyDetail.txt. To rename output or input files for the CIMSActt, CIMSSort, and CIMSBill programs, you need to include separate steps for each program rather than using a single step for the SingleProcessStep program.</p>

**Table 2-3 • SampleNightly.xml Job File Contents**

For a description of the Jobs and Job elements and attributes, see [page 2-48](#).

```
<?xml version="1.0" encoding="utf-8"?>
<Jobs xmlns="http://www.cimslab.com/CIMSJobs.xsd">
  <Job id="Nightly"
    description="Daily Collection"
    active="true"
    dataSourceId=""
    joblogShowStepParameters="true"
    joblogShowStepOutput="true"
    processPriorityClass="Low"
    joblogWriteToTextFile="true"
    joblogWriteToXMLFile="false"
    smtpSendJobLog="true"
    smtpServer="mail.cimslab.com"
    smtpFrom="CIMSPProcessResults@cimslab.com"
    smtpTo="CIMSPProcessResults@cimslab.com"
    stopOnProcessFailure="false">
```

For a description of the Process element and attributes, see [page 2-52](#).

```
<Process id="MSIIS-Web"
  description="Process for IIS Collection"
  joblogShowStepOutput="true"
  joblogShowStepParameters="true"
  active="true">
```

For a description of the Defaults and Default elements and attributes, see [page 2-90](#).

```
<Defaults>
  <Default programName="CIMSACCT"
    accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTab1-Win.txt"/>
</Defaults>
```

For a description of the Steps and Step elements and attributes, see [page 2-55](#).

```
<Steps stopOnStepFailure="true">
  <Step id="Server1 Collection"
    description="Server1 IIS"
    type="ConvertToCSR"
    programName="MSIIS\MSIIS.wsf"
    programType="wsf"
    active="true">
```

For a description of the valid parameters for each collector, refer to the collector-specific information in the following chapters.

```
<Parameters>
  <Parameter Feed="Server1"/>
  <Parameter LogFolder="\\Server1\Logfiles"/>
  <Parameter ProcessType="web"/>
  <Parameter SiteIDOrAll="All"/>
</Parameters>
</Step>

<Step id="Server2 Collection"
  description="Server2 IIS"
  type="ConvertToCSR"
  programName="MSIIS\MSIIS.wsf"
  programType="wsf"
  active="true">
  <Parameters>
    <Parameter Feed="Server2"/>
    <Parameter LogFolder="\\Server2\Logfiles"/>
    <Parameter ProcessType="web"/>
    <Parameter SiteIDOrAll="All"/>
  </Parameters>
</Step>
```

For descriptions of the parameters for the Scan program, see [page 2-62](#).

The Process step uses default parameters. To specify parameters for CIMSAct, CIMSSort, and CIMSBill, provide a separate step for each as shown in the SQL Server collector example on [page 2-34](#).

For descriptions of the parameters for the Cleanup program, see [page 2-85](#).

For a descriptions of the parameters for the FileTransfer program, see [page 2-80](#).

```

<Step id="Scan"
      description="Scan MSIIS"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
  <Parameters>
    <Parameter retainFileDate ="false"/>
    <Parameter allowMissingFiles="false"/>
    <Parameter allowEmptyFiles="false"/>
    <Parameter useStepFiles="false"/>
  </Parameters>
</Step>
<Step id="Process"
      description="Standard Processing for MSIIS"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
</Step>
<Step id="DatabaseLoad"
      description="Database Load for MSIIS"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
</Step>
<Step id="Cleanup"
      description="Cleanup MSIIS"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
    <Parameter CleanSubfolders="true"/>
  </Parameters>
</Step>
</Steps>
</Process>
<Process id="MSEExchange"
         description="Process for Exchange Server Collection"
         active="true">
  <Defaults>
    <Default programName="CIMSACCT"
             accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTabl-Win.txt"/>
  </Defaults>
  <Steps stopOnStepFailure="true">
    <Step id="FileTransfer 1"
          description="Transfer Exchange Server Log"
          type="Process"
          programName="FileTransfer"
          programType="net"
          active="true">
      <Parameters>
        <Parameter type="ftp"/>
        <Parameter serverName="ftp.xyzco.com"/>
        <Parameter userId="xyzco\billh"/>
        <Parameter userPassword="1234"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

```

        <Parameter from="ftp:///Exchsrvr/%LogDate_End%.log"
            to="file://\\Server1\LogFolder\Exchange1"
            action="Copy"
            overwrite="true"/>
    </Parameters>
</Step>
<Step id="FileTransfer 2"
    description="Transfer Exchange Server Log"
    type="Process"
    programName="FileTransfer"
    programType="net"
    active="true">
    <Parameters>
        <Parameter type="Windows"/>
        <Parameter from="\\Server3\Exchsrvr%\LogDate_end%.log"
            to="\\Server1\LogFolder\Exchange2"
            action="Copy"
            overwrite="true"/>
    </Parameters>
</Step>
<Step id="Server2 Collection"
    description="Server2 MExchange"
    type="ConvertToCSR"
    programName="MExchange\MExchange2003.wsf"
    programType="wsf"
    active="true">
    <Parameters>
        <Parameter Feed="Server2"/>
        <Parameter LogFolder="\\Server1\LogFolder\Exchange1"/>
    </Parameters>
</Step>
<Step id="Server3 Collection"
    description="Server3 MExchange"
    type="ConvertToCSR"
    programName="MExchange\MExchange2003.wsf"
    programType="wsf"
    active="true">
    <Parameters>
        <Parameter Feed="Server3"/>
        <Parameter LogFolder="\\Server1\LogFolder\Exchange2"/>
    </Parameters>
</Step>
<Step id="Scan"
    description="Scan MExchange"
    type="Process"
    programName="Scan"
    programType="net"
    active="true">
    <Parameters>
        <Parameter retainFileDate ="false"/>
        <Parameter allowMissingFiles="false"/>
        <Parameter allowEmptyFiles="false"/>
        <Parameter useStepFiles="false"/>
    </Parameters>
</Step>

```

For descriptions of the parameters for the DBLoad program, see [page 2-76](#).

```

        <Step id="Process"
            description="Standard Processing for MSEExchange"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
        </Step>
        <Step id="DatabaseLoad"
            description="Database Load for MSEExchange"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup MSEExchange"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
                <Parameter CleanSubfolders="true"/>
            </Parameters>
        </Step>
    </Steps>
</Process>
<Process id="Mainframe"
    description="Process for Mainframe Collection"
    active="true">
    <Defaults>
        <Default programName="CIMSACCT"
            accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTabl-Main.txt"/>
    </Defaults>
    <Steps stopOnStepFailure="true">
        <Step id="WaitForFile"
            description="Wait for CSR+ File"
            type="Process"
            programName="WaitFile"
            programType="net"
            active="true">
            <Parameters>
                <Parameter pollingInterval="60"/>
                <Parameter fileName="%ProcessFolder%\Host%\LogDate_End%.txt"/>
                <Parameter timeOutDateTime="%RNDATE% 13:50:59"/>
            </Parameters>
        </Step>
        <Step id="Scan"
            description="Scan Mainframe"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
            <Parameters>
                <Parameter retainFileDate ="false"/>
                <Parameter allowMissingFiles="false"/>
                <Parameter allowEmptyFiles="false"/>
                <Parameter useStepFiles="false"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

For descriptions of the parameters for the WaitFile program, see [page 2-78](#).

For a description of the %<value>% variables, see [page 2-61](#).

```

        </Parameters>
    </Step>
    <Step id="Process"
        description="Standard Processing for Mainframe"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>
    <Step id="DatabaseLoad"
        description="Database Load for Mainframe"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup Mainframe"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
            <Parameter CleanSubfolders="true"/>
        </Parameters>
    </Step>
</Steps>
</Process>
<Process id="UnixFS"
    description="Process for Unix Filesystem Collection"
    active="true">
    <Defaults>
        <Default programName="CIMSACCT"
            accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTab1-UNIX.txt"/>
    </Defaults>
    <Steps stopOnStepFailure="true">
        <Step id="Scan"
            description="Scan UnixFS"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
            <Parameters>
                <Parameter retainFileDate ="false"/>
                <Parameter allowMissingFiles="false"/>
                <Parameter allowEmptyFiles="false"/>
                <Parameter useStepFiles="false"/>
            </Parameters>
        </Step>
        <Step id="Prorate"
            description="Prorate CSR File"
            type="Process"
            programName="CIMSPRAT.exe"
            programType="console"
            active="true">
    
```

For descriptions of the parameters for the CIMSPrat program, see [page 2-64](#) and [page 2-87](#).

For information about proration, see [page 2-15](#).

```
<Parameters>
  <Parameter useStandardParameters="false"/>
  <Parameter useCommandProcessor="false"/>
  <Parameter XMLFileName="\\Server1\Prorate\CIMSPrat.xml"/>
</Parameters>
</Step>
<Step id="Process"
  description="Standard Processing for UnixFS"
  type="Process"
  programName="SingleProcessStep"
  programType="com"
  active="true">
</Step>
<Step id="DatabaseLoad"
  description="Database Load for UnixFS"
  type="Process"
  programName="DBLoad"
  programType="com"
  active="true">
</Step>
<Step id="Cleanup"
  description="Cleanup UnixFS"
  type="Process"
  programName="Cleanup"
  programType="net"
  active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
    <Parameter CleanSubfolders="true"/>
  </Parameters>
</Step>
</Steps>
</Process>
<Process id="MSSQL2000"
  description="Process for MSSQL2000 Collection"
  active="true">
  <Defaults>
    <Default programName="CIMSACCT"
      accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTab1-Win.txt"/>
    <!--NOTE: The first time that you run the job file for the SQL Server 2000
  collector, the job will fail if there are no existing files in the trace
  folder. However, a trace file with the run date in the file name is created
  as a result of the job run. If you want to ensure that the job file runs
  correctly, run the job file again with the RNDATE keyword.-->
    <!--Default logDate="RNDATE"/-->
  </Defaults>
  <Steps stopOnStepFailure="true">
    <Step id="Server 1 Collection"
      description="Server1 MSSQL2000"
      type="ConvertToCSR"
      programName="MSSQLServer\2000\MSSQL2000"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter TraceFolder="\\Server1\TraceFolder"/>
        <Parameter DataSourceID="CIMSServer"/>
        <Parameter RunSP="true"/>
      </Parameters>
```



For descriptions of the parameters for the CIMSacct program, see [page 2-65](#).

```

</Step>
<Step id="Scan"
description="Scan MSSQL2000"
type="Process"
programName="Scan"
programType="net"
active="true">
  <Parameters>
    <Parameter retainFileDate="false"/>
    <Parameter allowMissingFiles="false"/>
    <Parameter allowEmptyFiles="false"/>
    <Parameter useStepFiles="false"/>
  </Parameters>
</Step>
<Step id="CIMSACCT"
description="CIMSacct for MSSQL2000"
type="Process"
programName="CIMSacct"
programType="com"
processPriorityClass="BelowNormal"
active="true">
  <Parameters>
    <!--Parameter inputFile="CurrentCSR.txt"/-->
    <!--Parameter detailFile="Detail.txt"/-->
    <!--Parameter accCodeConvTable="AcctTbl.txt"/-->
    <!--Parameter resultsFile="AcctResults.txt"/-->
    <!--Parameter controlFile="AcctCntl.txt"/-->
    <!--Parameter messageFile="AcctMsg.txt"/-->
    <!--Parameter exceptionFile="Exception.txt"/-->
    <!--Parameter identFile="Ident.txt"/-->
    <!--Parameter createDBInf="true"/-->
    <!--Parameter createCSRFile="false"/-->
    <!--Parameter CSRFile="CSRFile.txt"/-->
    <!--Parameter controlCard="VERIFY DATA ON"/-->
    <!--Parameter controlCard="ACCOUNT FIELDS,Folder,1,24"/-->
    <!--Parameter logMessageFileOutput="true"/-->
    <!--Parameter logResultFileOutput="true"/-->
  </Parameters>

```

For descriptions of the parameters for the CIMSSort program, see [page 2-70](#).

```

</Step>
<Step id="CIMSSORT"
description="CIMSSort for MSSQL2000"
type="Process"
programName="CIMSSort"
programType="com"
active="true">
  <Parameters>
    <!--Parameter inputFilename="Detail.txt"/-->
    <!--Parameter outputFilename="Detail.txt"/-->
  </Parameters>

```

For descriptions of the parameters for the CIMSbill program, see [page 2-71](#).

```

</Step>
<Step id="CIMSBILL"
description="CIMSbill for MSSQL2000"
type="Process"
programName="CIMSbill"
programType="com"
active="true">
  <Parameters>
    <!--Parameter detailFileIn="Detail.txt"/-->
    <!--Parameter detailFileOut="BillDetail.txt"/-->
  </Parameters>

```

```

        <!--Parameter summaryFile="BillSummary.txt"/-->
        <!--Parameter resultsFile="BillResults.txt"/-->
        <!--Parameter controlFile="BillCntl.txt"/-->
        <!--Parameter messageFile="BillMsg.txt"/-->
        <!--Parameter multTableFile="" /-->
        <!--Parameter createDBInf="true"/-->
        <!--Parameter dateSelection="Date Keyword | YYYYMMDD"/-->
        <!--Parameter reportDate="Date Keyword | YYYYMMDD"/-->
        <!--Parameter controlCard="PROCESS DETAIL RECORDS"/-->
        <!--Parameter controlCard="DEFINE J1 1 1"/-->
        <!--Parameter logMessageFileOutput="true"/-->
        <!--Parameter logResultFileOutput="true"/-->
    </Parameters>
</Step>
<Step id="DatabaseLoad"
description="Database Load for MSSQL2000"
type="Process"
programName="DBLoad"
programType="com"
active="true">
</Step>
<Step id="Cleanup"
description="Cleanup MSSQL2000"
type="Process"
programName="Cleanup"
programType="net"
active="true">
    <Parameters>
        <Parameter DaysToRetainFiles="45"/>
        <Parameter cleanSubfolders="true"/>
    </Parameters>
</Step>
</Steps>
</Process>
<Process id="CIMSWinDisk"
description="Process for CIMS Windows Disk Collector"
active="true">
    <Defaults>
        <Default programName="CIMSACCT"
accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTabl-Win.txt"/>
        <Default LogDate="RNDATE"/>
    </Defaults>
    <Steps stopOnStepFailure="true">
        <Step id="Server1 Collection"
description="Server1 CIMSWinDisk"
type="ConvertToCSR"
programName="CIMSWinDisk\CIMSWinDisk.exe"
programType="console"
active="true">
            <GenerateExternalFile filename="%ProcessFolder%\CIMSWinDisk.xml"
overwrite="true">
                <![CDATA[
                    <?xml version="1.0" encoding="utf-8" ?>
                    <CIMSCollectors version = "1.0">
                    <Collectors>
                        <Collector name="CIMSWinDisk"
instanceName="Server1-C"
instanceDescription="Scan of Server1 C Drive"
Active="True">

```

For descriptions of the GenerateExternalFile element and attributes, see [page 2-59](#).

For a descriptions of the Collectors and Collector elements, attributes, and parameters for CIMSWinDisk, see [page 7-5](#).

```

        <Parameters>
            <Parameter name="LogDate" value="%RNDATE%" />
            <Parameter name="Retention" value="KEEP" />
            <Parameter name="Feed" value="Server1-C" />
            <Parameter name="OutputFolder"
                value="%ProcessFolder%" />
            <Parameter name="PathToScan"
                value="\\Server1\C$" />
            <Parameter name="Units" value="GB" />
            <Parameter name="NumberOfLevels" value="1" />
        </Parameters>
    </Collector>
</Collectors>
</CIMSCollectors>
]]>
</GenerateExternalFile>
<Parameters>
<!--IMPORTANT NOTE: If Smart Scan is enabled in the Scan step
(useStepFiles="true"), you must include a Feed parameter that exactly
matches the Feed in the GenerateExternalFile above or include the
scanFile="<file name>" parameter that specifies the path and file name of
the file to be scanned-->
<!--Parameter Feed=""/-->
<!--Parameter scanFile=""/-->
    <Parameter UseStandardParameters="false"/>
    <Parameter XMLFileName="%ProcessFolder%\CIMSWinDisk.xml" />
    <Parameter CollectorName="CIMSWinDisk"/>
</Parameters>
</Step>
<Step id="Scan"
    description="Scan CIMSWinDisk"
    type="Process"
    programName="Scan"
    programType="net"
    active="true">
    <Parameters>
        <Parameter retainFileDate ="false"/>
        <Parameter allowMissingFiles="false"/>
        <Parameter allowEmptyFiles="false"/>
        <Parameter useStepFiles="false"/>
    </Parameters>
</Step>
<Step id="Process"
    description="Standard Processing for CIMSWinDisk"
    type="Process"
    programName="SingleProcessStep"
    programType="com"
    active="true">
</Step>
<Step id="DatabaseLoad"
    description="Database Load for CIMSWinDisk"
    type="Process"
    programName="DBLoad"
    programType="com"
    active="true">
</Step>
<Step id="Cleanup"
    description="Cleanup CIMSWinDisk"
    type="Process"

```

```

        programName="Cleanup"
        programType="net"
        active="true">
    <Parameters>
        <Parameter DaysToRetainFiles="45"/>
        <Parameter CleanSubfolders="true"/>
    </Parameters>
</Step>
</Steps>
</Process>
<Process id="CIMSWinEventLog"
description="Process for CIMS WinEventLog Collection"
active="true">
    <Defaults>
        <Default programName="CIMSACCT"
accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTab1-Win.txt"/>
    </Defaults>
    <Steps stopOnStepFailure="true">
    <Step id="PrintSrvr Collection"
description="PrintSrvr CIMSWinEventLog"
type="ConvertToCSR"
programName="CIMSWinEventLog\CIMSWinEventLog.exe"
programType="console"
active="true">
    <GenerateExternalFile filename="%ProcessFolder%\CIMSWinEventLog.xml"
overwrite="true">
        <![CDATA[
            <?xml version="1.0" encoding="utf-8" ?>
            <CIMSCollectors version = "1.0">
                <Collectors>
                    <Collector name="CIMSWinEventLog"
instanceName="PrintSrvr"
instanceDescription="PrintSrvr Event Log"
Active="True">
                        <Parameters>
                            <Parameter name="LogDate" value="%LOGDATE%"/>
                            <Parameter name="Retention" value="KEEP" />
                            <Parameter name="Feed" value="PrintSrvr" />
                            <Parameter name="OutputFolder"
value="%ProcessFolder%" />
                            <Parameter name="LogSource"
value="\\PrintSrvr\EventLog\%LogDate_End%.evt"/>
                            <Parameter name="LogType" value="file" />
                            <Parameter name="EventType" value="Print" />
                        </Parameters>
                    </Collector>
                </Collectors>
            </CIMSCollectors>
        ]]>
    </GenerateExternalFile>
    <Parameters>
    <!--IMPORTANT NOTE: If Smart Scan is enabled in the Scan step
(useStepFiles="true"), you must include a Feed parameter that exactly
matches the Feed in the GenerateExternalFile above or include the
scanFile="<file name>" parameter that specifies the path and file name of
the file to be scanned-->
    <!--Parameter Feed=""/-->
    <!--Parameter scanFile=""/-->
        <Parameter UseStandardParameters="false"/>
    </Parameters>

```

For descriptions of the GenerateExternalFile element and attributes, see [page 2-59](#).

For a descriptions of the Collectors and Collector elements, attributes, and parameters for CIMSWinEventLog, see [page 9-5](#).

```

        <Parameter XMLFileName="%ProcessFolder%\CIMSWinEventLog.xml" />
        <Parameter CollectorName="CIMSWinEventLog" />
    </Parameters>
</Step>
    <Step id="Scan"
        description="Scan CIMSWinEventLog"
        type="Process"
        programName="Scan"
        programType="net"
        active="true">
        <Parameters>
            <Parameter retainFileDate ="false" />
            <Parameter allowMissingFiles="false" />
            <Parameter allowEmptyFiles="false" />
            <Parameter useStepFiles="false" />
        </Parameters>
    </Step>
    <Step id="Process"
        description="Standard Processing for CIMSWinEventLog"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>
    <Step id="DatabaseLoad"
        description="Database Load for CIMSWinEventLog"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup CIMSWinEventLog"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45" />
            <Parameter CleanSubfolders="true" />
        </Parameters>
    </Step>
</Steps>
</Process>
<Process id="DBSpace"
    description="Process for DBSpace Collection"
    active="true">
    <Defaults>
        <Default programName="CIMSACCT"
            accCodeConvTable="C:\CIMS\AccountCodeTable\AcctTab1-Win.txt" />
        <Default LogDate="RNDATE" />
    </Defaults>
    <Steps stopOnStepFailure="true">
        <Step id="Server1 Collection"
            description="Server1 DBSpace"
            type="ConvertToCSR"
            programName="DBSpace\DBSpace.wsf"
            programType="wsf"
            active="true">

```

## ■ Installing CIMS Data Collectors and Setting Up the System

### Setting Up the System

Note that this process contains separate steps for CIMSAct CIMSBill, and CIMSSort

```
<Parameters>
  <Parameter Feed="Server1"/>
  <Parameter DBType="MS"/>
  <Parameter DataSourceID="CIMSServer"/>
</Parameters>
</Step>
<Step id="Scan"
  description="Scan DBSpace"
  type="Process"
  programName="Scan"
  programType="net"
  active="true">
  <Parameters>
    <Parameter retainFileDate ="false"/>
    <Parameter allowMissingFiles="false"/>
    <Parameter allowEmptyFiles="false"/>
    <Parameter useStepFiles="false"/>
  </Parameters>
</Step>
<Step id="CIMSACCT"
  description="CIMSAct for DBSpace"
  type="Process"
  programName="CIMSAct"
  programType="com"
  processPriorityClass="BelowNormal"
  active="true">
  <Parameters>
    <Parameter identFile="MyIdent.txt"/>
  </Parameters>
</Step>
<Step id="CIMSSORT"
  description="CIMSSort for DBSpace"
  type="Process"
  programName="CIMSSort"
  programType="com"
  active="true">
</Step>
<Step id="CIMSBILL"
  description="CIMSBill for DBSpace"
  type="Process"
  programName="CIMSBill"
  programType="com"
  active="true">
  <Parameters>
    <Parameter detailFileOut="MyDetail.txt"/>
  </Parameters>
</Step>
<Step id="DatabaseLoad"
  description="Database Load for DBSpace"
  type="Process"
  programName="DBLoad"
  programType="com"
  active="true">
  <Parameters>
    <Parameter loadType="BillDetail" fileName="MyDetail.txt"/>
    <Parameter loadType="Ident" fileName="MyIdent.txt"/>
  </Parameters>
</Step>
```

```
<Step id="Cleanup"
      description="Cleanup DBSpace"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
    <Parameter CleanSubfolders="true"/>
  </Parameters>
</Step>
</Steps>
</Process>
</Job>
</Jobs>
```

## Example of Reprocessing With Smart Scan Enabled

One or more ConvertToCSR steps in a job might fail (for example, a log file that matches the LogDate parameter was not produced, network connection was lost and the CSR file was not sent to the feed subfolder, etc.). In this situation, you can correct the problem that caused the steps to fail and then create a job file to reprocess just the failed steps.

Using the SampleNightly.xml job file as an example, assume that the ConvertToCSR steps for IIS on Server 2 (see page 2-29) and CIMSWinEventLog on Server 1 (see page 2-38) failed. All other ConvertToCSR steps in the job file ran successfully.

Rather than running the entire SampleNightly.xml file again, you could create a second job file as follows:

```
<Jobs xmlns="http://www.cimslab.com/CIMSJobs.xsd">
  <Job id="Nightly Reprocess"
    description="Daily Collection"
    active="true"
    dataSourceId=""
    joblogShowStepParameters="true"
    joblogShowStepOutput="true"
    processPriorityClass="Low"
    joblogWriteToTextFile="true"
    joblogWriteToXMLFile="false"
    smtpSendJobLog="true"
    smtpServer="mail.cimslab.com"
    smtpFrom="CIMSPProcessResults@cimslab.com"
    smtpTo="CIMSPProcessResults@cimslab.com"
    stopOnProcessFailure="false">
    <Process id="MSIIS-Web"
      description="Process for IIS Collection"
      joblogShowStepOutput="true"
      joblogShowStepParameters="true"
      active="true">
      <Defaults>
        <Default programName="CIMSACCT"
          accCodeConvTable="C:\CIMS\AccountCodeTable\AccTab1-Win.txt"/>
      </Defaults>
      <Steps stopOnStepFailure="true">
        <Step id="Server2 Collection"
          description="Server2 IIS"
          type="ConvertToCSR"
          programName="MSIIS\MSIIS.wsf"
          programType="wsf"
          active="true">
          <Parameters>
            <Parameter Feed="Server2"/>
            <Parameter LogFolder="\\Server2\LogFiles"/>
            <Parameter ProcessType="web"/>
            <Parameter SiteIDOrAll="All"/>
          </Parameters>
        </Step>
        <Step id="Scan"
          description="Scan MSIIS"
          type="Process"
          programName="Scan"
          programType="net"
          active="true">

```

Here you reprocess the IIS ConvertToCSR step that failed.



The useStepFiles="true" attribute enables Smart Scan. See [page 2-45](#).

```

        <Parameters>
            <Parameter retainFileDate ="false"/>
            <Parameter allowMissingFiles="false"/>
            <Parameter allowEmptyFiles="false"/>
            <Parameter useStepFiles="true"/>
        </Parameters>
    </Step>
    <Step id="Process"
        description="Standard Processing for MSIIS"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>
    <Step id="DatabaseLoad"
        description="Database Load for MSIIS"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup MSIIS"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>
<Process id="CIMSWinEventLog"
    description="Process for CIMSWinEventLog Collector"
    active="true">
    <Defaults>
        <Default programName="CIMSACCT"
            accCodeConvTable="C:\CIMS\AccountCodeTable\AccTab1-Win.txt"/>
    </Defaults>
    <Steps stopOnStepFailure="true">
    <Step id="PrintSrvr Collection"
        description="PrintSrvr CIMSWinEventLog"
        type="ConvertToCSR"
        programName="CIMSWinEventLog\CIMSWinEventLog.exe"
        programType="console"
        active="true">
        <GenerateExternalFile filename="%ProcessFolder%\CIMSWinEventLog.xml"
            overwrite="true">
            <![CDATA[
                <?xml version="1.0" encoding="utf-8" ?>
                <CIMSCollectors version = "1.0">
                    <Collectors>
                        <Collector name="CIMSWinEventLog"
                            instanceName="PrintSrvr"
                            instanceDescription="PrintSrvr Event Log"
                            Active="True">

```

Here you reprocess the CIMSWinEvent Log ConvertToCSR step that failed.

```

        <Parameters>
            <Parameter name="LogDate" value="%LOGDATE%" />
            <Parameter name="Retention" value="KEEP" />
            <Parameter name="Feed" value="PrintSrvr" />
            <Parameter name="OutputFolder"
                value="%ProcessFolder%" />
            <Parameter name="LogSource"
                value="\\PrintSrvr\LogFiles\LogDate_End%.evt"/>
            <Parameter name="LogType" value="file" />
            <Parameter name="EventType" value="Print" />
        </Parameters>
    </Collector>
</Collectors>
</CIMSCollectors>
]]>
</GenerateExternalFile>
<Parameters>
    <Parameter UseStandardParameters="false" />
    <Parameter XMLFileName="%ProcessFolder%\CIMSWinEventLog.xml" />
    <Parameter CollectorName="CIMSWinEventLog" />
    <Parameter scanFile="\\Server1\CIMSWinEventLog\PrintSrvr\
        %LogDate_End%.txt" />
</Parameters>
</Step>
<Step id="Scan"
    description="Scan CIMSWinEventLog"
    type="Process"
    programName="Scan"
    programType="net"
    active="true">
    <Parameters>
        <Parameter retainFileDate ="false" />
        <Parameter allowMissingFiles="false" />
        <Parameter allowEmptyFiles="false" />
        <Parameter useStepFiles="true" />
    </Parameters>
</Step>
<Step id="Process"
    description="Standard Processing for CIMSWinEventLog"
    type="Process"
    programName="SingleProcessStep"
    programType="com"
    active="true">
</Step>
<Step id="DatabaseLoad"
    description="Database Load CIMSWinEventLog"
    type="Process"
    programName="DBLoad"
    programType="com"
    active="true">
</Step>
<Step id="Cleanup"
    description="Cleanup CIMSWinEventLog"
    type="Process"
    programName="Cleanup"
    programType="net"
    active="true">
    <Parameters>
        <Parameter DaysToRetainFiles="45" />

```

The scanFile= attribute or a Feed attribute is required when processing CIMSWinEventLog of CIMWinDisk. See [page 2-45](#).

```
        </Parameters>
      </Step>
    </Steps>
  </Process>
</Job>
</Jobs>
```

The parameter attribute `useStepFiles="true"` enables the Smart Scan feature. When Smart Scan is on, an entry is made in an internal table each time a `ConvertToCSR` step in the process is run successfully. The table entry is a file name entered in the format *process definition folder\feed subfolder\LogDate.txt* (for example, `MSIIS-Web\Server2\20050916.txt`). Smart Scan will search for only those file names defined in the table and will ignore all other feed subfolders and files in the process definition folder.

The Smart Scan feature uses the `Feed` parameter values to determine the files to be scanned. For most collectors, the `Feed` parameter is provided in the `ConvertToCSR` step(s) in the job file. However, the `Feed` parameter for the CIMS Windows Disk and CIMS Windows Event Log collectors is provided in an external file. To use Smart Scan with these collectors, you need to include either of the following in the `ConvertToCSR` step(s):

- A `Feed` parameter.

Or

- A `scanFile="file name"` parameter where the file name includes the full path of the file to be scanned as shown in the preceding job file example. For a description of the `scanFile` parameter, see [page 2-89](#).

## Job File Structure

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

---

**Note** • If the same attribute is included for more than one element in the job file, the value in the lowest element takes precedence. For example, if an attribute is defined in the `Jobs` element and the child `Job` element, the value for the `Job` element attribute takes precedence.

---

## Jobs Element

The `Jobs` element is the root element of the job file. All other elements are child elements of `Jobs`.

Table 2-4 lists the attributes for the `Jobs` element. These attributes are optional. The SMTP attributes enable you to send the logs generated for all jobs in the job file via one e-mail message. You can also use these attributes to send a separate e-mail message for each individual job (see *Job Element* on page 2-48). These attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Attribute	Required or Optional	Description
<code>processFolder</code>	Optional	In most cases, you will not need to use this attribute. By default, the path to the <code>Processes</code> folder set in the <code>CIMSConfigOptions</code> table is used.  This attribute is required only if you are collecting data on a computer other than the central CIMS Data Collectors server and you cannot access the other computer from the central computer. (For example, the second computer is behind a firewall). This attribute enables you to "pull" CSR files from the other computer to the central server for processing.
<code>smtpSendJobLog</code>	Optional	Specifies whether the job log should be sent via e-mail. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (send via e-mail)</li> <li>■ "false" (do not send)</li> </ul> The default is "false".
<code>smtpServer</code>	Optional	The name of the SMTP mail server that will be used to send the job log.  The default is "mail.cimslab.com".

**Table 2-4 • Jobs Element Attributes**

Attribute	Required or Optional	Description
smtpFrom	Optional	<p>The fully qualified e-mail address of the e-mail sender.</p> <p>The default is "cims.server@cimslab.com".</p>
smtpTo	Optional	<p>The fully qualified e-mail address of the e-mail receiver.</p> <p>The syntax for an address defined by this attribute can be any of the following.</p> <ul style="list-style-type: none"> <li>■ user@domain                     <p>Example: jsmith@xyzco.com</p> <p>When this syntax is used, the default mail server is the server defined by the smtpServer attribute.</p> </li> <li>■ servername:user@domain                     <p>Example: mail.xyzco.com:jsmith@xyzco.com</p> <p>When this syntax is used, the mail server specified for this attribute overrides the server defined by the smtpServer attribute.</p> </li> <li>■ servername:userID:password:user@domain                     <p>Example: mail.xyzco.com:janes:global:jsmith@xyzco.com</p> </li> <li>■ servername:userID:password:port:user@domain                     <p>Example: mail.xyzco.com:janes:global:25:jsmith@xyzco.com</p> </li> </ul> <p>If you want to use multiple addresses, separate them with a semicolon (;). You can use any combination of address syntaxes in a multiple address list. For example, "jsmith@xyzco.com;mail.pdqco.com:bhughes@pdqco.com".</p> <p>The default is "CIMSProcessResults@cimslab.com".</p>

Table 2-4 • Jobs Element Attributes (Continued)

**Job Element**

XML tree structure: Jobs/Job

A Job element starts the definition of a job within the job file. A job is composed of one or more processes that run specific data collectors.

You can define multiple jobs in the job file. For example, you might have a job named *Nightly* that includes all data collectors that you want to run nightly and another job named *Monthly* that includes all collectors that you want to run monthly.

Table 2-5 lists the attributes for the Job element. Some optional attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Attribute	Required or Optional	Description
id	Required	<p>A text string name for the job. This value must be unique from other job ID values in the file.</p> <p><b>Example:</b></p> <pre>id="Nightly"</pre> <p>In this example, the subfolder that contains log files for this job will also be named <i>Nightly</i>. See <i>Job Log Files (LogFiles Folder)</i> on page 2-8.</p>
description	Optional	<p>A text string description of the job (maximum of 255 characters).</p> <p><b>Example:</b></p> <pre>description="Nightly collection and processing"</pre>
active	Optional	<p>Specifies whether the job should be run. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (run the job)</li> <li>■ "false" (do not run the job)</li> </ul> <p>The default is "true".</p>

**Table 2-5 • Job Element Attributes**

Attribute	Required or Optional	Description
dataSourceId	Optional	<p>The CIMS Data Source for the CIMS Server database.</p> <p><b>Example:</b> dataSourceId=CSDev</p> <p>If this parameter is not provided, the CIMS Data Source that is set as the Web/collector default in the CIMS Data Source Maintenance dialog box in CIMS Server Administrator is used.</p> <p>To use a CIMS Data Source other than the default, set this parameter to the appropriate CIMS Data Source ID.</p> <p>For more information about creating and using CIMS Data Sources, see <a href="#">Appendix B, CIMS Data Sources</a>.</p>
joblogShowStepParameters	Optional	<p>Specifies whether parameters for the steps in a job are written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (parameters are written to the job log)</li> <li>■ "false" (parameters are not written)</li> </ul> <p>The default is "true".</p>
joblogShowStepOutput	Optional	<p>Specifies whether output generated by the steps in a job is written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (step output is written to the job log)</li> <li>■ "false" (step output is not written)</li> </ul> <p>The default is "true".</p>

**Table 2-5 • Job Element Attributes (Continued)**

Attribute	Required or Optional	Description
processFolder	Optional	<p>In most cases, you will not need to use this attribute. By default, the path to the Processes folder set in the CIMSSConfigOptions table is used.</p> <p>This attribute is required only if you are collecting data on a computer other than the central CIMS Data Collectors server and you cannot access the other computer from the central computer. (For example, the second computer is behind a firewall). This attribute enables you to "push" CSR files from the other computer to the central server for processing.</p>
processPriorityClass	Optional	<p>Determines the priority in which the job is run. Valid values are: Low, BelowNormal (the default), Normal, AboveNormal, and High. Because a job can use a large amount of CPU time, CIMS Lab recommends that you use the Low or BelowNormal value, which allows other processes (for example, IIS and SQL Server tasks) to take precedence. Consult CIMS Lab before using a value other than Low or BelowNormal.</p> <p><b>Note:</b> A priority of Low or BelowNormal will not cause the job to run longer if the system is idle. However, if other tasks are running, the job will take longer.</p>
joblogWriteToTextFile	Optional	<p>Specifies whether the job log should be written to a text file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (writes to a text file)</li> <li>■ "false" (does not write to a text file)</li> </ul> <p>The default is "true".</p>
joblogWriteToXMLFile	Optional	<p>Specifies whether the job log should be written to an XML file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (writes to an XML file)</li> <li>■ "false" (does not write to an XML file)</li> </ul> <p>The default is "false".</p>

**Table 2-5 • Job Element Attributes (Continued)**



Attribute	Required or Optional	Description
smtpSendJobLog	Optional	Specifies whether the job log should be sent via e-mail. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (send via e-mail)</li> <li>■ "false" (do not send)</li> </ul> The default is "false".
smtpServer	Optional	The name of the SMTP mail server that will be used to send the job log. The default is "mail.cimslab.com".
smtpFrom	Optional	The fully qualified e-mail address of the e-mail sender. The default is "cims.server@cimslab.com".
smtpTo	Optional	The fully qualified e-mail address of the e-mail receiver. See the description on <a href="#">page 2-47</a> .
stopOnProcessFailure	Optional	Specifies whether a job with multiple processes should stop if any of the processes fail. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (stop processing)</li> <li>■ "false" (continue processing)</li> </ul> The default is "false".

---

**Table 2-5 • Job Element Attributes (Continued)**

**Process Element**

XML tree structure: Jobs/Job/Process

A Process element starts the definition of a data collection process within a job. A job can contain multiple process elements.

A process defines the type of data collected (IIS, SQL Server 2000, operating system, print, etc.).

Table 2-6 lists the attributes for the Process element. Some optional attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Attribute	Required or Optional	Description
id	Required	<p>A text string name for the process. This value must be unique from the other process ID values in the job.</p> <p>This value must match the name of a process definition folder for a collector in the Processes folder (see <i>Process Definitions (Processes Folder)</i> on page 2-12).</p> <p>If the buildProcessFolder attribute is not included or is set to "true" (the default), CIMS Job Runner will create a process definition folder of the same name in the Processes folder if the process definition folder does not exist.</p> <p><b>Example:</b></p> <pre>id="ABCSoftware"</pre> <p>In this example, the process definition folder created by CIMS Job Runner will be named ABCSoftware.</p>
description	Optional	<p>A text string description of the process (maximum of 255 characters).</p> <p><b>Example:</b></p> <pre>description="Process for ABCSoftware"</pre>

**Table 2-6 • Process Element Attributes**

Attribute	Required or Optional	Description
buildProcessFolder	Optional	<p>Specifies whether CIMS Job Runner will create a process definition folder with the same name as the <code>id</code> attribute value in the <code>Processes</code> folder.</p> <p>If you are using CIMS Job Runner to perform data collection, a process folder is always required. If you do not include this attribute or set it to <code>"true"</code>, a process definition folder is created automatically if it does not already exist.</p> <p>This attribute is only applicable if you are using CIMS Job Runner to run a script or program that does not require a process definition folder. For example, you can use CIMS Job Runner to run the <code>ReportDistribution.wsf</code> script used for batch reporting as described in the <i>CIMS Server Administrator's Guide</i>.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ <code>"true"</code> (the process definition folder is created)</li> <li>■ <code>"false"</code> (the process definition folder is not created)</li> </ul> <p>The default is <code>"true"</code>.</p>
joblogShowStepParameters	Optional	<p>Specifies whether parameters for the steps in a process are written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ <code>"true"</code> (parameters are written to the job log)</li> <li>■ <code>"false"</code> (parameters are not written)</li> </ul> <p>The default is <code>"true"</code>.</p>
joblogShowStepOutput	Optional	<p>Specifies whether output generated by the steps in a process is written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ <code>"true"</code> (step output is written to the job log)</li> <li>■ <code>"false"</code> (step output is not written)</li> </ul> <p>The default is <code>"true"</code>.</p>

Table 2-6 • Process Element Attributes (Continued)

Attribute	Required or Optional	Description
processPriorityClass	Optional	<p>This attribute determines the priority in which the process is run. Valid values are: Low, BelowNormal (the default), Normal, AboveNormal, and High. Because a process can use a large amount of CPU time, CIMS Lab recommends that you use the Low or BelowNormal value, which allows other processes (for example, IIS and SQL Server tasks) to take precedence. Consult CIMS Lab before using a value other than Low or BelowNormal.</p> <p><b>Note:</b> A priority of Low or BelowNormal will not cause the process to run longer if the system is idle. However, if other tasks are running, the process will take longer.</p>
active	Optional	<p>Specifies whether the process should be run. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (run the process)</li> <li>■ "false" (do not run the process)</li> </ul> <p>The default is "true".</p>

**Table 2-6 • Process Element Attributes (Continued)**

## Steps Element

XML tree structure: Jobs/Job/Process/Steps

A Steps element is a container for one or more Step elements. The Steps element has one optional attribute as shown in Table 2-7.

Attribute	Required or Optional	Description
stopOnStepFailure	Optional	<p>Specifies whether processing should continue if any of the active steps in the process fail. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (processing fails)</li> <li>■ "false" (processing continues)</li> </ul> <p>The default is "true".</p>

**Table 2-7 • Steps Element Attribute**

## Step Element

XML tree structure: Jobs/Job/Process/Steps/Step

A Step element defines a step within a process.

---

**Note •** A Step element can occur at the process level or the job level.

---

Table 2-8 lists the attributes for the Step element. Some optional attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Attribute	Required or Optional	Description
id	Required	<p>A text string name for the step. This value must be unique from other step ID values in the process.</p> <p><b>Example:</b></p> <pre>id="Scan"</pre> <p>In this example, the step is executing the Scan program.</p>
description	Optional	<p>A text string description of the step (maximum of 255 characters).</p> <p><b>Example:</b></p> <pre>description="Scan ABCSOFTWARE"</pre>

**Table 2-8 • Step Element Attributes**

Attribute	Required or Optional	Description
active	Optional	<p>Specifies whether the step should be run. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (run the step)</li> <li>■ "false" (do not run the step)</li> </ul> <p>The default is "true".</p>
type	Required	<p>The type of step that is being implemented: "ConvertToCSR" or "Process".</p> <p>"ConvertToCSR" specifies that the step performs data collection and conversion and creates a CSR file.</p> <p>Process specifies that the step executes one of the programs described in <i>CIMS Processing Programs (Process Engine\JobLibrary Folder)</i> on page 2-9.</p>
programName	Required	<p>The name of the program that will be run by the step.</p> <p>If the type attribute is ConvertToCSR and the programType attribute is wsf, this value can be either of the following:</p> <ul style="list-style-type: none"> <li>■ The path and name of a conversion script in the Collectors folder. For example, "MSEExchange\MSEExchange2000.wsf" specifies the conversion script for the Exchange 2000 Server collector.</li> <li>■ The name of the script in the Scripts folder. If the script is in a subfolder of the Scripts folder, you need to include the path. For example, Batch Reporting\ReportDistribution.wsf.</li> </ul>

**Table 2-8 • Step Element Attributes (Continued)**

Attribute	Required or Optional	Description
programName (continued)	Required	<p>If the <code>type</code> attribute is <code>ConvertToCSR</code> and the <code>programType</code> attribute is <code>console</code>, this value can be the full path or just the name of console application (make sure that you include the file extension, e.g., <code>CIMSPRAT.exe</code>).</p> <p>If you do not include the path, CIMS Job Runner searches the <code>Collectors</code>, <code>Process Engine</code>, and <code>Scripts</code> folders for the program.</p> <p>If the <code>type</code> attribute is <code>Process</code>, this value is the name of a CIMS program (e.g., <code>"Scan"</code>, <code>"CIMSACCT"</code>, <code>"CIMSBILL"</code>, <code>"DBLoad"</code>, etc).</p> <p><b>Examples:</b></p> <pre>programName="MSIIS\MSIIS.wsf" programName="CIMSWinDisk.exe" programName="Cleanup"</pre>
processPriorityClass	Optional	<p>This attribute determines the priority in which the step is run. Valid values are: <code>Low</code>, <code>BelowNormal</code> (the default), <code>Normal</code>, <code>AboveNormal</code>, and <code>High</code>. Because a step can use a large amount of CPU time, CIMS Lab recommends that you use the <code>Low</code> or <code>BelowNormal</code> value, which allows other processes (for example, IIS and SQL Server tasks) to take precedence. Consult CIMS Lab before using a value other than <code>Low</code> or <code>BelowNormal</code>.</p> <p><b>Note:</b> A priority of <code>Low</code> or <code>BelowNormal</code> will not cause the step to run longer if the system is idle. However, if other tasks are running, the step will take longer.</p>

Table 2-8 • Step Element Attributes (Continued)

Attribute	Required or Optional	Description
programType	Optional	<p>The type of program specified by the programName attribute:</p> <ul style="list-style-type: none"> <li>■ "wsf"—Windows Scripting File</li> <li>■ "ce"—CIMS Conversion Engine</li> <li>■ "console"—Console Application</li> <li>■ "com"—COM Component</li> <li>■ "net"—.Net Component</li> </ul> <p>The default is "net".</p>
joblogShowStepParameters	Optional	<p>Specifies whether parameters for the step are written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (parameters are written to the job log)</li> <li>■ "false" (parameters are not written)</li> </ul> <p>The default is "true".</p>
joblogShowStepOutput	Optional	<p>Specifies whether output generated by the step is written to the job log file. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (step output is written to the job log)</li> <li>■ "false" (step output is not written)</li> </ul> <p>The default is "true".</p>

**Table 2-8 • Step Element Attributes (Continued)**



## GenerateExternalFile Element

XML tree structure: Jobs/Job/Process/Steps/Step/GenerateExternalFile

A GenerateExternalFile element defines a file that is generated before the step is run. For example, the CIMS Windows Disk and CIMS Event Log collectors require an XML file that provides the parameters for these collectors. You can define these files within the job file so that you do not have to create and maintain separate files. For a description of these files, see *Defining the CIMSWinDisk.xml File* on page 7-5 and *Defining the CIMSWinEventLog.xml File* on page 9-5.

Table 2-9 lists the attributes for the GenerateExternalFile element. Some optional attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Attribute	Required or Optional	Description
fileName	Required	The name of the file to be generated. A full path is optional. If you do not provide the full path, the file is created in the process definition folder for the collector.  <b>Note:</b> If you provide a full path, the path must be an existing path unless you include the attribute createPath="true".
overwrite	Optional	Specifies whether the file should overwrite an existing file. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (the existing file is overwritten)</li> <li>■ "false" (the file is not overwritten and the step fails)</li> </ul> The default is "true".

**Table 2-9 • GenerateExternalFile Element Attributes**

Attribute	Required or Optional	Description
autoRemove	Optional	<p>Specifies whether the file should be automatically removed after the step has executed. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the file is removed)</li> <li>■ "false" (the file is not removed)</li> </ul> <p>The default is "false".</p>
createPath	Optional	<p>This attribute works in conjunction with the <code>fileName</code> attribute. If you include a full path for <code>fileName</code>, but the path does not exist, this attribute specifies whether the path is automatically created. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the path is created)</li> <li>■ "false" (the path is not created)</li> </ul> <p>The default is "false".</p>

**Table 2-9 • GenerateExternalFile Element Attributes (Continued)**

**Parameters Element**

XML tree structure: Jobs/Job/Process/Steps/Parameters

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

**Parameter Element**

XML tree structure: Jobs/Job/Process/Steps/Parameters/Parameter

A Parameter element defines a parameter to a step.

The valid attributes for conversion step parameters (`type=ConvertToCSR`) depend on the collector called by the step. For the parameters/attributes required for a specific collector, refer to the section describing that collector. Individual collectors are described in [Chapter 3](#) through [Chapter 13](#)

The valid attributes for process step parameters (`type=Process`) are listed in [Table 2-10](#) on page 2-62. The attributes are broken down as follows

- Parameter attributes that are specific to a program (Scan, CIMSacct, CIMSBill, etc.) begin on [page 2-62](#).
- Parameter attributes that are specific to a program type (wsf, com, net, console, etc.) begin on [page 2-87](#).

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 used by CIMSAct and CIMSBill, do not include the path with the file name. These programs should reside in the collector's process definition folder.

The exceptions are the account code conversion table used by CIMSAct ([page 2-65](#)) and the proration table used by CIMSBill (see [page 2-72](#)). You can place these files in a central location so that they can be used by multiple processes. In this case, you need to provide the path.

- Attributes include macro capability so that the following pre-defined strings, as well as environment strings, will automatically be expanded at run time.
  - %ProcessFolder%. Specifies the Processes folder as defined in the CIMSCfgOptions table or by the processFolder attribute.
  - %LogDate%. Specifies that the LogDate parameter value is to be used.
  - %<Date Keyword>%. Specifies that a date keyword (RNDATE, CURMON, PREMON, etc.) is to be used.
  - %LogDate\_End%. For files that contain a date, specifies that files that contain a date matching the last day of the LogDate parameter value are used. For example, if the LogDate parameter value is CURMON, files with dates for the last day of the current month are used. For single day values such as PREDAY, the start and end date are the same.
  - %LogDate\_Start%. For files that contain a date, specifies that files that contain a date matching the first day of the LogDate parameter value are used. For example, if the LogDate parameter value is CURMON, files with dates for the first day of the current month are used. For single day values such as PREDAY, the start and end date are the same.

Program Name or Type	Attribute	Required or Optional	Description
<b>Valid Parameters by Program Name</b>			
Scan  For a description of this program, see <a href="#">page 2-9</a> .  For an example of the parameters for this program in a job file, see <a href="#">page 2-30</a> .	retainFileDate	Optional	Specifies whether the date is retained in the final CSR file (i.e., <i>yyyymmdd.txt</i> rather than <i>CurrentCSR.txt</i> ). Valid values are: <ul style="list-style-type: none"> <li>■ "true" (the file name is <i>yyyymmdd.txt</i>)</li> <li>■ "false" (the file name is <i>CurrentCSR.txt</i>)</li> </ul> The default is "false".
	allowMissingFiles	Optional	Specifies whether a warning or error occurs when feed subfolders do not contain a file that matches the log date value. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (a warning occurs, processing continues)</li> <li>■ "false" (an error occurs, processing fails)</li> </ul> The default is "false".
	allowEmptyFiles	Optional	Specifies whether a warning or error occurs when feed subfolders contain a zero-length file that matches the log date value. Valid values are: <ul style="list-style-type: none"> <li>■ "true" (a warning occurs, processing continues)</li> <li>■ "false" (an error occurs, processing fails)</li> </ul> The default is "false".

**Table 2-10 • Parameter Element Attributes**

Program Name or Type	Attribute	Required or Optional	Description
Scan (continued)	excludeFile	Optional	<p>The name of a file to be excluded from the Scan process. The file can be in any feed subfolder in the collector's process definition folder. The file name can include wildcard characters but not a path.</p> <p><b>Example:</b></p> <pre>excludeFile="MyCSR*"</pre> <p>In this example, all files that begin with MyCSR are not scanned.</p>
	excludeFolder	Optional	<p>The name of a feed subfolder in to be excluded from the Scan process. The subfolder name can include wildcard characters but not a path. The feed subfolder must be a top-level folder within the process definition folder. (For more information about the feed subfolder, see <a href="#">page 2-13</a>).</p> <p><b>Example:</b></p> <pre>excludeFolder="Server1"</pre> <p>In this example, the feed subfolder Server1 is not scanned.</p>
	includeFile	Optional	<p>The name of a file to be included in the Scan process. Files with any other name will be excluded from the Scan process. Include a path if the file is in a location other than a feed subfolder in collector's process definition folder.</p> <p><b>Example:</b></p> <pre>includeFile="MyCSR.txt"</pre> <p>In this example, files in the feed subfolders that are named MyCSR are scanned.</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
Scan (continued)	useStepFiles	Optional	<p>Specifies whether the Smart Scan feature is enabled (see <a href="#">page 2-42</a> for a description of this feature). Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (Smart Scan is enabled)</li> <li>■ "false" (Smart Scan is not enabled)</li> </ul> <p>The default is "false".</p> <p>By default, Smart Scan looks for a file named <i>LogDate.txt</i> in the process definition feed subfolders (e.g., MExchange/Server1/20050624.txt). If you want to override the default name, use the parameter attribute <i>scanFile</i> in the conversion step (see <a href="#">page 2-89</a>).</p>
CIMSPRAT For a description of this program, see <a href="#">page 2-10</a> .	XMLFileName	Required	The path and the name of the XML file containing the proration parameters used by CIMSPrat.
For an example of the parameters for this program in a job file see <a href="#">page 2-33</a> .	useStandardParameters and useCommandProcessor	Optional	For descriptions of these attributes, see <a href="#">page 2-87</a> .

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CIMSACCT  For a description of this program, see page 2-9.	inputFile	Optional	<p>The name of the CSR or CSR+ file to be processed. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>inputFile="MyCSR.txt"</pre> <p>The default is "CurrentCSR.txt".</p> <p><b>Note:</b> The input file can also be a CIMSacct Detail file or CIMS Summary file. These files are usually processed through CIMSacct to perform further account code conversion. For more information about account code conversion, refer to the <i>CIMS Server Administrator's Guide</i>.</p>
	detailFile	Optional	<p>The name of the CIMSacct Detail file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>detailFile="MyDetail.txt"</pre> <p>The default is "Detail.txt".</p>
	accCodeConvTable	Optional	<p>The name of account code conversion table used by CIMSacct. Include a path if the table is in a location other than the collector's process definition folder.</p> <p><b>Examples:</b></p> <pre>accCodeConvTable="MyAcctTbl.txt"</pre> <pre>accCodeConvTable="E:\Processes\Account\MyAcctTbl.txt"</pre> <p>The default is "AcctTbl.txt".</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
CIMSACCT (continued)	resultsFile	Optional	<p>The name of the CIMSacct Results file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>resultsFile= "MyAcctResults.txt"</pre> <p>The default is "AcctResults.txt".</p>
	controlFile	Optional	<p>The name of the control file used by CIMSacct. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>controlFile= "MyAcctCntl.txt"</pre> <p>The default is "AcctCntl.txt".</p>
	messageFile	Optional	<p>The name of the CIMSacct Message file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>messageFile= "MyAcctMsg.txt"</pre> <p>The default is "AcctMsg.txt".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**



Program Name or Type	Attribute	Required or Optional	Description
CIMSACCT (continued)	exceptionFile	Optional	<p>The name of the exception file produced by CIMSacct. This file must be in the collector's process definition folder—do not include a path.</p> <p>The file name should contain the log date so that it is not overwritten when CIMSacct is run again.</p> <p><b>Example:</b></p> <pre>exceptionFile= "Exception_%LogDate_End%. txt"</pre> <p>The default is "Exception.txt".</p>
	identFile	Optional	<p>The name of the CIMS Ident file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>identFile="MyIdent.txt"</pre> <p>The default is "Ident.txt".</p>
	createDBInf	Optional	<p>Specifies whether the ODBCINF.txt file should be generated. This file is required by the Fujitsu COBOL Workstation Run-time program required by CIMS Processing Engine. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the file is generated)</li> <li>■ "false" (the file is not generated)</li> </ul> <p>The default is "true".</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
CIMSACCT (continued)	controlCard	Optional	<p>A valid CIMSacct control statement or statements. All CIMSacct control statements are stored in the CIMSacct control file (see <a href="#">page 2-66</a>).</p> <p><b>Note:</b> If you have an existing CIMSacct control file in the process definition folder, the statements that you define as controlCard parameters will overwrite all statements currently in the file.</p> <p>To define multiple control statements, you need to use a separate parameter for each statement.</p> <p><b>Example:</b></p> <pre>&lt;parameter controlCard="TEST A" /&gt; &lt;parameter controlCard="VERIFY DATA ON" /&gt;</pre>
	logMessageFileOutput	Optional	<p>Specifies whether the text of the CIMSacct Message file is included in the job log file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the text is included)</li> <li>■ "false" (the text is not included)</li> </ul> <p>The default is "true".</p>
	logResultFileOutput	Optional	<p>Specifies whether the text of the CIMSacct Results file is included in the job log file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the text is included)</li> <li>■ "false" (the text is not included)</li> </ul> <p>The default is "true".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CIMSACCT (continued)	createCSRPFfile	Optional	<p>Specifies whether a CSR+ file is written. CSR+ files are the same as CSR files except the file records contain an additional record header (see <i>Chapter 11, Mainframe Data Collectors</i>).</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (CSR+ is written)</li> <li>■ "false" (CSR+ is not written)</li> </ul> <p>The default is "false".</p>
	CSRPFfile	Optional	<p>Specifies the name of the CSR+ file that will be written.</p> <p><b>Example:</b></p> <p>CSRPFfile="MyCSRPFfile.txt"</p> <p>The default is "CSRPFfile.txt".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CIMSSORT  For a description of this program, see <a href="#">page 2-10</a> .	inputFilename	Optional	<p>The file name for the CIMSacct Detail file.</p> <p>The path is required only if the file is not in the process definition folder.</p> <p>Examples:</p> <pre>inputFileName="Detail.txt" inputFileName="E:\CIMS\Detail.txt"</pre> <p>The default is &lt;default process definition folder path&gt; +"Detail.txt"</p>
	outputFilename	Optional	<p>The path and file name for the sorted output.</p> <p>The path is required only if the file is not sent to the process definition folder.</p> <p>Examples:</p> <pre>outputFileName="Detail.txt" outputFileName="E:\CIMS\Detail.txt"</pre> <p>The default is &lt;default process definition folder path&gt; +"Detail.txt"</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CIMSBILL  For a description of this program, see page 2-10.	detailFileIn	Optional	<p>The name of the input CIMSacct Detail, CIMSBill Detail, or CIMS Summary file to be processed. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>detailFileIn="MyDetail.txt"</pre> <p>The default is "Detail.txt".</p>
	detailFileOut	Optional	<p>The name of the CIMSBill Detail file produced. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>detailFileOut="MyBillDetail.txt"</pre> <p>The default is "BillDetail.txt".</p>
	summaryFile	Optional	<p>The name of the CIMS Summary file produced. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>summaryFile="MyBillSummary.txt"</pre> <p>The default is "BillSummary.txt".</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
CIMSBILL (continued)	resultsFile	Optional	<p>The name of the CIMSBill Results file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>resultsFile= "MyBillResults.txt"</pre> <p>The default is "BillResults.txt".</p>
	controlFile	Optional	<p>The name of the control file used by CIMSBill. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>controlFile= "MyBillCntl.txt"</pre> <p>The default is "BillCntl.txt".</p>
	messageFile		<p>The name CIMSBill Message file. This file must be in the collector's process definition folder—do not include a path.</p> <p><b>Example:</b></p> <pre>messageFile= "MyBillMsg.txt"</pre> <p>The default is "BillMsg.txt".</p>
	multTableFile	Optional	<p>The name of the proration table used by CIMSBill. Include a path if the table is in a location other than the collector's process definition folder.</p> <p><b>Examples:</b></p> <pre>multTableFile= "MyMultTable.txt"</pre> <pre>multTableFile= "E:\Processes\Prorate\ MyMultTable.txt"</pre>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CIMSBILL (continued)	createDBInf	Optional	<p>Specifies whether the ODBCINF.txt file should be generated. This file is required by the Fujitsu COBOL Workstation Run-time program required by CIMS Processing Engine. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the file is generated)</li> <li>■ "false" (the file is not generated)</li> </ul> <p>The default is "true".</p>
	dateSelection	Optional	<p>Defines a date range for records to be processed by CIMS Bill. Valid values are a from and to date range in yyyyymmdd format or a CIMS date keyword.</p> <p><b>Examples:</b></p> <p>dateSelection="20050117 20050118"</p> <p>In this example, CIMS Bill will process records with an accounting end dates of January 17 and 18, 2005.</p> <p>dateSelection="PREDAY"</p> <p>In this example, CIMS Bill will process records with an accounting end date one day prior to the date CIMS Job Runner is run. For more information about accounting dates, refer to the <i>CIMS Server Administrator's Guide</i>.</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
CIMSBILL (continued)	reportDate	Optional	<p>Defines the dates that are used as the accounting start and end dates in the Summary records created by CIMSBill. Valid values are a date in yyyyymmdd format or a CIMS date keyword.</p> <p>You will not need to change the accounting dates for most chargeback situations. An example of a use for this feature is chargeback for a contractor's services for hours worked in the course of a month. In this case, you could set a report date of "CURMON", which sets the accounting start date to the first of the month and the end date to the last day of the month.</p>
	controlCard	Optional	<p>A valid CIMSBill control statement. All CIMSBill control statements are stored in the CIMSBill control file (see <a href="#">page 2-72</a>).</p> <p><b>Note:</b> If you have an existing CIMSBill control file in the process definition folder, the statements that you define as controlCard parameters will overwrite all statements currently in the file.</p> <p>To define multiple control statements, you need to use a separate parameter for each statement.</p> <p><b>Example:</b></p> <pre>&lt;parameter controlCard= "CLIENT SEARCH ON" /&gt;  &lt;parameter controlCard= "DEFINE J1 1 1" /&gt;</pre>

**Table 2-10 • Parameter Element Attributes (Continued)**



Program Name or Type	Attribute	Required or Optional	Description
CIMSBILL (continued)	logMessageFileOutput	Optional	<p>Specifies whether the text of the CIMS Bill Message file is included in the job log file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the text is included)</li> <li>■ "false" (the text is not included)</li> </ul> <p>The default is "true".</p>
	logResultFileOutput	Optional	<p>Specifies whether the text of the CIMS Bill Results file is included in the job log file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the text is included)</li> <li>■ "false" (the text is not included)</li> </ul> <p>The default is "true".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
DBLoad  For a description of this program, see <a href="#">page 2-10</a> .  For an example of the parameters for this program in a job file see <a href="#">page 2-30</a> .	loadType	Required if you want to load a specific file rather all files.	By default, the DBLoad program loads the Summary, CIMSBill Detail, and Ident files into the database.  If you want to load a specific file rather than all files, the valid values are: <ul style="list-style-type: none"> <li>■ Summary</li> <li>■ BillDetail (Detail file produced by CIMSBill)</li> <li>■ AcctDetail (Detail file produced by CIMSAct)</li> <li>■ Ident</li> </ul> <p>For more information about these file types, refer to the <i>CIMS Server Administrator's Guide</i>.</p>
	filename	Required if loadType attribute is used	The file name for the file to be loaded. If the file is in a location other than the collector's process definition folder, you need to include the path.  The default file names are: <ul style="list-style-type: none"> <li>■ BillSummary.txt or BillSummary_yyyymmdd.txt (if the file was produced by CIMS Mainframe)</li> <li>■ BillDetail.txt or BillDetail_yyyymmdd.txt</li> <li>■ Ident.txt or Ident_yyyymmdd.txt</li> </ul> <p><b>Example:</b> file="BillSummary.txt"</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
DBLoad (continued)	allowDetailDuplicat	Optional	<p>Specifies whether duplicate Detail files can be loaded into the database. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (duplicate loads can be loaded)</li> <li>■ "false" (duplicate loads cannot be loaded)</li> </ul> <p>The default is "false".</p>
	allowSummaryDuplicat	Optional	<p>Specifies whether duplicate Summary files can be loaded into the database. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (duplicate loads can be loaded)</li> <li>■ "false" (duplicate loads cannot be loaded)</li> </ul> <p>The default is "false".</p>
	useBulkLoad	Optional	<p>Specifies whether the SQL Server bulk load facility should be used to improve load performance. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (bulk load is used)</li> <li>■ "false" (bulk load is not used)</li> </ul> <p>The default is "true".</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
DBLoad (continued)	useDatedFiles	Optional	<p>Summary, Detail, and Ident files produced by CIMS Mainframe include the date in the file name. For example, BillSummary_ <i>yyyymmdd</i>.txt.</p> <p>If set to "true", only files that contain a date matching the LogDate parameter value are loaded into the database. The default is "false".</p> <p><b>Note:</b> Summary, Detail, and Ident files with dates in the file name are produced only by CIMS Mainframe Data Collector and Chargeback System 12.0 and later. For all other collectors, dates are not included in these file names.</p>
WaitFile  For a description of this program, see <a href="#">page 2-11</a> .  For an example of the parameters for this program in a job file see <a href="#">page 11-3</a> and <a href="#">page 11-5</a> .	pollingInterval	Optional	<p>The number of seconds to check for file availability (maximum of 10,080 [one week]).</p> <p><b>Example:</b></p> <pre>pollingInterval="60"</pre> <p>This example specifies a polling interval of 60 seconds.</p> <p>The default is 5 seconds.</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
WaitFile (continued)	timeout	Optional	<p>The number of seconds that CIMS Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails.</p> <p><b>Example:</b></p> <pre>timeout="18000"</pre> <p>This example specifies a timeout of 5 hours.</p> <p>The default is to wait indefinitely.</p>
	timeoutDateTime	Optional	<p>A date and time up to which CIMS Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails.</p> <p>The date and time must be in the format <code>yyymmdd hh:mm:ss</code>.</p> <p><b>Example:</b></p> <pre>timeoutDateTime="%rndate% 23:59:59"</pre> <p>This example specifies a timeout of 23:59:59 on the day CIMS Job Runner is run.</p> <p>The default is to wait indefinitely.</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
WaitFile (continued)	filename	Required	<p>The name of the file to wait for. If a path is not specified, the path to the process definition folder for the collector is used. The file must be available before the step can continue.</p> <p>If the file contains a date, for example files produced by CIMS Mainframe Data Collector and Chargeback System, include a variable string for the date.</p> <p><b>Example:</b></p> <pre>filename="BillSummary_ %LogDate_End%.txt"</pre> <p>In this example, CIMS Job Runner will wait for CIMS Mainframe Summary files that contain the same end date as the %LogDate_End% value (see <a href="#">page 2-61</a>).</p>
FileTransfer	continueOnError	Optional	<p>For a multi-file transfer, specifies whether subsequent file transfers continue if a transfer fails. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (file transfer continues)</li> <li>■ "false" (file transfer does not continue)</li> </ul> <p>The default is "false".</p>
	type	Required	<p>The type of file transfer. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "ftp" (File Transfer Protocol [<i>FTP</i>] transfer)</li> <li>■ "Windows" (Windows transfer)</li> </ul>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
FileTransfer (continued)	The following attributes from, to, action, and overwrite are attributes of a single Parameter element. If you are transferring multiple files, include a Parameter element with these attributes for each file.		
	For an example of these attributes in a job file, see <a href="#">page 2-30</a> .		
	from and to	Required	<p>The location of the source file and the destination file. The values that you can enter for these attributes are dependent on the type attribute value as follows:</p> <ul style="list-style-type: none"> <li>■ type="ftp" <p>Specify the from and to file paths as shown in the following examples. The examples differ depending on whether you are transferring the file from or to a ftp server. The ftp server is specified by the serverName attribute (see <a href="#">page 2-84</a>).</p> <pre>from="ftp:///LogFiles/ %LogDate_End%.log" to=file:// \\Server1\LogFiles\ %LogDate_End%.log" or from=file:// \\Server1\LogFiles\ %LogDate_End%.log" to="ftp:///LogFiles/ %LogDate_End%.log"</pre> <p>Note that the use of a UNC is recommended for the file:// path as shown in these examples.</p> <p>The from and to file names can be different.</p> <p>For a description of the %LogDate_End% variable (see <a href="#">page 2-61</a>).</p> </li> </ul>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
FileTransfer (continued)	from and to	Required	<p>■ type="Windows"</p> <p>You can include the URL prefix file:// before the the from and to file paths or leave it off. The use of a UNC path is recommended as shown in the following example:</p> <pre>from="file:// \\Server1\LogFiles\ %LogDate_End%.log" to="file:// \\Server2\LogFiles\ %LogDate_End%.log"</pre> <p>The from and to file names can be different.</p> <p>The file name in the from path can contain wildcards. If wildcards are included, do not include the file name in the to path as shown in the following example:</p> <pre>from="\\Server1\LogFiles \%LogDate_End%*.log" to="\\Server2\LogFiles"</pre>
	action	Required	<p>Specifies the file activity. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "Copy" (copies the file from the from location to the to location)</li> <li>■ "Delete" (deletes the file from the from location)</li> <li>■ "Move" (copies the file from the from location to the to location and then deletes the file from the from location)</li> </ul> <p>The default is Copy.</p>

**Table 2-10 • Parameter Element Attributes (Continued)**



Program Name or Type	Attribute	Required or Optional	Description
FileTransfer (continued)	overwrite	Optional	<p>Specifies whether the destination file is overwritten. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the file is overwritten)</li> <li>■ "false" (the file is not overwritten)</li> </ul> <p>The default is "false".</p>
<p>The following attributes are for FTP transfer only.</p>			
	connectionType	Optional	<p>Describes how the connection address is resolved. This is an advanced configuration option that should be used only after consulting CIMS Lab (see <i>Chapter 15, Contacting Technical Support</i>).</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "PRECONFIG" (retrieves the proxy or direct configuration from the registry)</li> <li>■ "DIRECT" (resolves all host names locally)</li> <li>■ "NOAUTOPROXY" (retrieves the proxy or direct configuration from the registry and prevents the use of a startup Microsoft JScript or Internet Setup (INS) file)</li> <li>■ "PROXY" (passes requests to the proxy unless a proxy bypass list is supplied and the name to be resolved bypasses the proxy)</li> </ul> <p>The default is "PRECONFIG".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
FileTransfer (continued)	passive	Optional	Forces the use of FTP passive semantics. In passive mode FTP, the client initiates both connections to the server. This solves the problem of firewalls filtering the incoming data port connection to the FTP client from the FTP server.  This is an advanced configuration option that should be used only after consulting CIMS Lab.
	proxyServerBypass	Optional	This is a pointer to a null-terminated string that specifies an optional comma-separated list of host names, IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. This option is used only when <code>connectionType="PROXY"</code> .  This is an advanced configuration option that should be used only after consulting CIMS Lab.
	proxyServer	Optional	If <code>connectionType="PROXY"</code> , the name of the proxy server(s) to use.  This is an advanced configuration option that should be used only after consulting CIMS Lab.
	serverName	Required	A valid FTP IP address or server name.  Example: <code>serverName="ftp.xyzco.com"</code>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
FileTransfer (continued)	transferType	Optional	<p>The type of file transfer. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "binary"</li> <li>■ "ascii"</li> </ul> <p>The default is "binary".</p>
	userId	Optional	The user ID used to log on to the FTP server.
	userPassword	Optional	The user password used to log on to the FTP server.
Cleanup	folder	Optional	<p>By default, the Cleanup program deletes files with file names containing the date in yyyyymmdd format from the collector's process definition folder.</p> <p>If you want to delete files from another folder, use this attribute to specify the path and folder name.</p> <p><b>Example:</b></p> <pre>folder="\\Server1\LogFiles"</pre>
	daysToRetainFiles	Optional	<p>The the number of days that you want to keep the yyyyymmdd files after their creation date.</p> <p><b>Example:</b></p> <pre>daysToRetainFiles="60"</pre> <p>This example specifies that all files that are older than 60 days from the current date are deleted.</p> <p>The default is 45 days from the current date.</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

## ■ Installing CIMS Data Collectors and Setting Up the System

### Setting Up the System

Program Name or Type	Attribute	Required or Optional	Description
Cleanup (continued)	dateToRetainFiles	Optional	<p>A date by which all yyyyymmdd files that were created prior to this date will be deleted. You can use a CIMS date keyword or the date in yyyyymmdd format.</p> <p><b>Example:</b></p> <pre>dateToRetainFiles="PREMON"</pre> <p>This example specifies that all files that were created prior to the previous month will be deleted.</p>
	cleanSubfolders	Optional	<p>Specifies whether the files that are contained in subfolders are deleted. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the files are deleted)</li> <li>■ "false" (the files are not deleted)</li> </ul> <p>The default is "false".</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
<b>Valid Parameters by Program Type</b>			
CONSOLE  (For an example of the parameters for this program in a job file see <a href="#">page 7-3.</a> )	useStandardParameters	Optional	<p>Specifies that if the program type is <code>console</code>, the standard parameters required for all conversion scripts are passed on the command line in the following order:</p> <ul style="list-style-type: none"> <li>■ LogDate</li> <li>■ RetentionFlag</li> <li>■ Feed</li> <li>■ OutputFolder</li> </ul> <p>These parameters are passed before any other parameters defined for the step. For more information about the standard parameters, see <a href="#">page 2-7.</a></p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the standard parameters are passed)</li> <li>■ "false" (the standard parameters are not passed)</li> </ul> <p>If the step type is <code>Process</code>, the default value is "false". If the step type is <code>ConvertToCSR</code>, the default is "true".</p>

Table 2-10 • Parameter Element Attributes (Continued)

Program Name or Type	Attribute	Required or Optional	Description
CONSOLE (continued)	useCommandProcessor	Optional	<p>Specifies whether the Cmd.exe program should be used to execute a console program. If the Cmd.exe program is not used, then the console program is called using APIs.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the Cmd.exe program is used)</li> <li>■ "false" (the Cmd.exe program is not used)</li> </ul> <p>The default is "true".</p>
	XMLFileName, CollectorName, and CollectorInstance	Optional	<p>These parameters are used by the CIMS Windows Disk and CIMS Windows Event Log collectors. They specify the name of the XML file used by the collector; the name of the collector; and the collector instance, respectively.</p> <p>For more information about these collectors, see <a href="#">Chapter 7</a> and <a href="#">Chapter 9</a>.</p>

**Table 2-10 • Parameter Element Attributes (Continued)**

Program Name or Type	Attribute	Required or Optional	Description
CONSOLE or WSF	scanFile	Optional	<p>This attribute is applicable only if the Smart Scan feature is enabled (see <a href="#">page 2-42</a> for more information about Smart Scan).</p> <p>When Smart Scan is enabled, the Scan program searches for CSR files that are defined in an internal table. The default path and name for these files is <i>process definition folder\feed subfolder\LogDate.txt</i>.</p> <p>If the file name to be scanned is other than the default defined in the table, you can use this attribute to specify the file name. You need to include the path as shown in the following example:</p> <pre>scanFile="\\Server1\MSIIS-Web\Server2\MyFile.txt"</pre> <p>If Smart Scan is enabled, you can also use this attribute to disable the scan of CSR files created by a particular CONSOLE or WSF step by specifying <code>scanFile=""</code> (empty string).</p>

Table 2-10 • Parameter Element Attributes (Continued)

#### Defaults Element (Optional)

XML tree structure: Jobs/Job/Process/Defaults

A Defaults element is a container for individual Default elements. The use of Default elements is optional.

#### Default Element (Optional)

XML tree structure: Jobs/Job/Process/Defaults/Default

A Default element defines a global value for a job or process. This element enables you to define parameter for multiple steps in one location as shown in the example on [page 2-91](#). There are two types of attributes that you can use in a Default element: pre-defined and user defined. [Table 2-11](#) lists the attributes for the Default element by type.

---

**Note** • If the same attribute appears in both a Default element for a job or process and a Parameter element for a step, the value in the Parameter element overrides the value in the Default element.

---

Attribute	Description
<b>Pre-defined attributes. These are the attributes that are pre-defined by CIMS Lab.</b>	
LogDate	The log date specifies the date for the data that you want to collect. You should enter the log date in the job file only if you are running a snapshot collector or the Transactions collector (see <a href="#">Using Log Dates in the Job File</a> on page 2-26).
RetentionFlag	This attribute is for future use. Valid values are KEEP or DISCARD.
<b>User-defined attributes. You can define additional default values using the following attributes. For an example of the use of these attributes, see <a href="#">Default Example</a> on page 2-91.</b>	
programName	A default can apply to a specific program or all programs in a job or process. If the default applies to a specific program, this attribute is required to define the program.
attribute name and value	The name of the attribute that you want to use as the default followed by a literal value for the attribute. The attribute name cannot contain spaces.

**Table 2-11 • Default Element Attributes**



## Default Example

This example contains two Default elements.

The first Default element is at the job level. This element specifies that all steps in the Nightly job that execute the CIMSacct program will use the same account code conversion table, ACCTTABL-WIN.txt, which is located in the specified path.

The second Default element is at the process level for the DBSpace collector. This element specifies that the DBSpace collector will be run using the log date RNDATE.

```
<Job      id="Nightly"
         description="Nightly collection and processing"
         active="true"
         processPriorityClass="BelowNormal"
         joblogWriteToTextFile="true"
         joblogWriteToXMLFile="true"
         joblogShowStepOutput="false"
         joblogShowStepParameters="false"
         smtpSendJobLog="true"
         smtpServer="mail.xyzco.com"
         smtpFrom="abcserver@xyzco.com"
         smtpTo="jmith@xyzco.com;mail.pdqco.com:bhughes@pdqco.com"
         stopOnProcessFailure="false">
  <Defaults>
    <Default  programName="CIMSACCT"
             accCodeConvTable="C:\CIMS\AccountCodeTable\ACCTTABL-WIN.txt"/>
  </Defaults>
  <Process  id="DBSpace"
           description="Process for DBSpace Collection"
           active="true">
    <Defaults>
      <Default  LogDate="RNDATE"/>
    </Defaults>
    <Steps>
      :
      :
```

## Running CIMS Data Collectors

This section describes how to use CIMS Job Runner to execute the data collection process. You should determine the frequency that you want to run CIMS Data Collectors as described in *Data Processing Frequency* before you run CIMS Job Runner.

### 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 (z/OS, UNIX, Windows, Open VMS, 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 manpower accounting, voice telephone, dedicated lines, etc., are normally 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. A single mid-sized Proxy Server might produce millions of records each day. 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 Folder Permissions for Data Processing

The administrator that executes processing using CIMS Data Collectors requires full access to files in the Processes folder (that is, the ability to create, modify, delete, overwrite, etc.). Therefore, the Windows user account or group account for the administrator must have Full Control security permissions for the Processes folder and all subfolders.

## Running CIMS Job Runner

You can run CIMS Job Runner (CIMSJobRunner.exe) directly from the command prompt or you can use Windows Task Scheduler to schedule the program to run automatically.

### *To run CIMS Job Runner from Task Scheduler:*

---

**Note** • These instructions are for Windows 2000 Server.

---

- 1 In Windows Control Panel, double-click **Scheduled Tasks**.
- 2 Double-click **Add Scheduled Tasks**.
- 3 The **Scheduled Task Wizard** appears. Click **Next**.
- 4 Click **Browse**, select the CIMSJobRunner.exe program, and then click **Next**.
- 5 Type a name for the task or accept the default and click the schedule for the task. Click **Next**.
- 6 Select the time and day to start the task, and then click **Next**.
- 7 Type the password for the user account on which you want the scheduled task to run. The password cannot be blank. Click **Next**.
- 8 Select the **Open advanced properties for this task when I click Finish** check box, and then click **Finish**.
- 9 In the **Task** tab, type the command and any parameters that you want to pass to CIMS Job Runner in the **Run** box as shown in the following example:  

```
"C:\Program Files\CIMSLab\Process Engine\CIMSJobRunner.exe" Nightly.xml
```

In this example, the job file Nightly.xml is specified as a parameter. For a list of other valid parameters and examples, see [Optional Parameters](#) on page 2-94.
- 10 Click **OK**.
- 11 In the **Set Account Information** dialog box, type the password for the user account again, and then click **OK**.

The task appears in the **Scheduled Task** list. To execute CIMSJobRunner.exe immediately, right-click the task, and then click **Run**. For more information about Task Scheduler, refer to the Microsoft documentation.

---

**Note** • CIMS Lab recommends that you create one job file for data collection. If you have multiple job files, you must schedule a separate instance of CIMS Job Runner to run each file. If the job files run concurrently, CIMS Server will attempt to load the output files from each job file into the database simultaneously, which might result in errors.

---

#### **To run CIMS Job Runner from the command prompt:**

At the `C:\Program Files\CIMSLab\Process Engine>` prompt type `cimsjobrunner.exe` followed by the optional parameters described in [Optional Parameters](#) on page 2-94. Or from any prompt, type `"C:\Program Files\CIMSLab\Process Engine\CIMSJobRunner.exe"` followed by the optional parameters.

#### **Examples**

```
C:\Program Files\CIMSLab\Process Engine>CIMSJobRunner.exe Nightly.xml
```

Or

```
C:\>"C:\Program Files\CIMSLab\Process Engine\CIMSJobRunner.exe" Nightly.xml
```

To ensure that the jobs within a job file run correctly, you might want to run the job file from the command line before using Windows Task Scheduler to run the job file. If an error occurs and the job(s) within a job file are not run (for example, the job file contains a syntax error) a job log file is not created and e-mail notification of the job failure is not sent.

#### **Optional Parameters**

You can supply the following optional parameters for CIMS Job Runner:

```
<job file name> <job id> <process id> <step id> <date literal | keyword>
```

Where:

`job file name` = the XML job file for the collector.

`job id` = the ID of a specific job in the job file that you want to run. The default is to run all jobs in the job file.

`process id` = the ID of a specific process that you want to run. If you include the `job id` parameter, the process applies only to that job. If you specify `All` as the `job id` parameter, the process applies to all jobs in the job file. The default is to run all processes in the job file.

`step id` = the ID of a specific step that you want to run. If you include the `process id` parameter, the step applies only to that process. If you specify `All` as the `process id` parameter, the step applies to all processes in the job file. The default is to run all steps in the job file.

`date literal | keyword` = a date literal or a CIMS date keyword. This parameter specifies the date for the data that you want to collect. If you do not provide a log date, the default date is the previous day. This is the equivalent of using the CIMS date keyword `PREDAY`.

For more information about using a log date, including valid log date values, see [Using Log Dates in the Job File](#) on page 2-26.

## Examples

CimsJobRunner.exe Nightly.xml	CIMS Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml.
CimsJobRunner.exe Nightly.xml Nightly	CIMS Job Runner runs all active processes and steps for the Nightly job in the job file Nightly.xml. No other jobs in the job file are run.
CimsJobRunner.exe Nightly.xml Nightly All DatabaseLoad	CIMS Job Runner runs only the active DatabaseLoad steps in all processes in the Nightly job. No other steps in the job file are run.
CimsJobRunner.exe Nightly.xml All All All 20050604	CIMS Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the LogDate parameter 20050604.
CimsJobRunner.exe Nightly.xml All All All RNDATE	CIMS Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the LogDate parameter RNDATE.

## ■ Installing CIMS Data Collectors and Setting Up the System

---

*Running CIMS Data Collectors*

# Operating System Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for operating systems. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Windows Process Data Collector</b> .....	<b>3-2</b>
Creating a Log On User Account for the CIMS Windows Process Collector Service (Optional) .	3-2
System Configuration Options for the CIMS Windows Process Collector .....	3-4
Installing the CIMS Windows Process Collector .....	3-9
Enabling CIMS Windows Process Logging .....	3-11
CIMS Windows Process Collector Log File Format .....	3-13
Identifiers and Resources Collected From the CIMS Windows Process Collector Log File .....	3-18
Setting Up the CIMS Windows Process Collector .....	3-19
Running the CIMS Windows Process Collector .....	3-24
<b>Windows System Resource Manager (WSRM) Collector</b> .....	<b>3-25</b>
Identifiers and Resources Collected by the WSRM Collector .....	3-25
Setting Up the WSRM Collector .....	3-26
<b>Citrix Data Collector</b> .....	<b>3-28</b>
Identifiers and Resources Collected by the Citrix Collector .....	3-28
Setting Up the Citrix Collector .....	3-29
Running the Citrix Collector .....	3-31
<b>VMware Data Collector</b> .....	<b>3-31</b>
Identifiers and Resources Collected by the VMware Collector .....	3-31
Setting Up the VMware Collector .....	3-32
Running the VMware Collector .....	3-34
<b>AS/400 Data Collector</b> .....	<b>3-34</b>

## Windows Process Data Collector

The CIMS Windows Process collector gathers usage data for processes running on Windows 2000/2003, XP, and NT operating systems and produces a log file of the data (see [CIMS Windows Process Collector Log File Format](#) on page 3-13). This log file provides useful metrics such as:

- Name and path of the process.
- Name of the computer that the process ran on.
- Name of the user that created the process.
- The elapsed CPU time used by the process (cumulative and broken down by kernel and user time).
- Bytes read and written by the process.

The following sections begin with important reference information for using the CIMS Windows Process collector, and then provide instructions for installing the collector, enabling process logging, and setting up and running the collector.

### Creating a Log On User Account for the CIMS Windows Process Collector Service (Optional)

The CIMS Windows Process collector includes a Windows *service* that supports the collector. By default, the service runs under the Local System user account. CIMS Lab recommends that you use this default account; however, you can run the service using a user or group account that has been granted the following security policies:

- Debug programs
- Log on as a service

You can assign these policies to a local account or a domain account. If you use a local account, you need to set the policies at both the domain and local level if you are using a domain. Policies for the domain override local policies.

#### Assigning Policies at the Domain Level

The following are the steps needed to assign these policies at the domain level. These steps assume that you are using a domain and Active Directory and that you have already created the local or domain account:

- 1 Open Active Directory Users and Computers (**Start** ▶ **Control Panel** ▶ **Administrative Tools** ▶ **Active Directory Users and Computers**).
- 2 In the Active Directory Users and Computers window, right-click the domain that you want, and then click **Properties**.
- 3 In the Properties dialog box, click the **Group Policy** tab, and then click **Edit**.
- 4 In the Group Policy window, navigate to **Computer Configuration** ▶ **Windows Settings** ▶ **Security Settings** ▶ **Local Policies** ▶ **User Rights Assignment**.



- 5 Double-click **User Rights Assignment**.
- 6 Double-click **Debug programs** and add the local or domain account in the Debug programs Properties dialog box. Make sure that the **Define these policy settings** check box is selected.
- 7 Double-click **Log on as a service** and repeat the procedures in the previous step.

### **Assigning Policies at the Local Level**

If you created a local account for the CIMS Windows Process service, complete the following steps.

- 1 From Control Panel, navigate to **Administrative Tools ▶ Local Security Policy ▶ Local Policies ▶ User Rights Assignment**.
- 2 Double-click **User Rights Assignment**.
- 3 Double-click **Debug programs** and add the local account in the Debug programs Properties dialog box.
- 4 Double-click **Log on as a service** and repeat the procedure in the previous step.

If you are using NTFS, make sure that the account has the right to access the folder where the CIMS Windows Process collector log file is written (see [Enabling CIMS Windows Process Logging](#) on page 3-11).

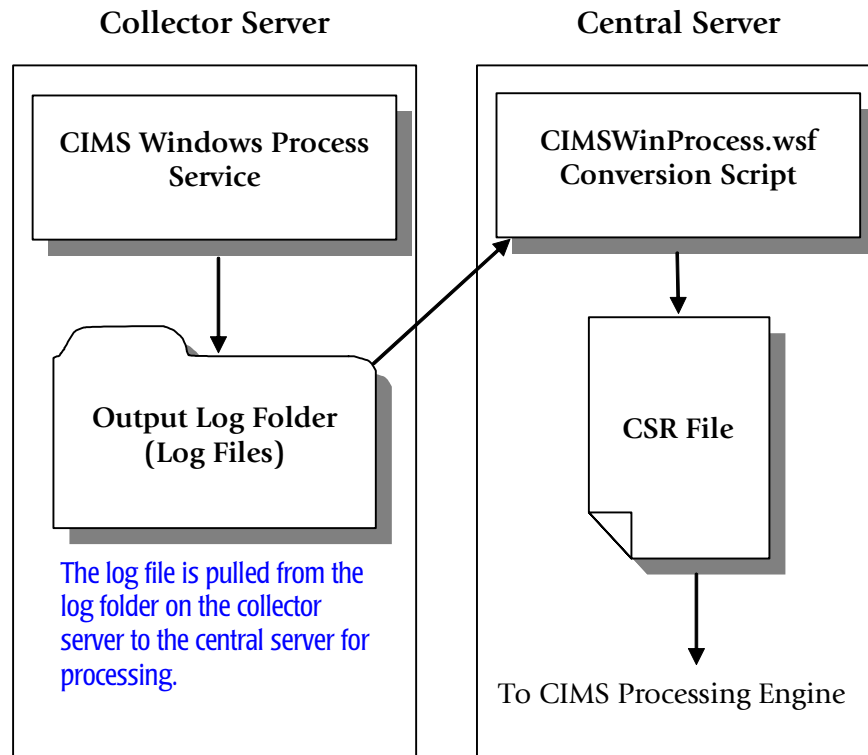
If the CIMS Windows Process collector is currently running under Local System, modify the service to run under the appropriate account.

## System Configuration Options for the CIMS Windows Process Collector

You can use any of the following system configurations to process the log files produced by the CIMS Windows Process collector. These configurations are presented in order of recommendation. The first configuration is the most simple and secure.

### Configuration 1: Pulling Log Files to the Central Server

In this configuration, the log files are written to a log folder on the server running the CIMS Windows Process collector and then pulled to the central CIMS Data Collectors server for processing.



**Figure 3-1 • System Configuration 1**

For an example of the job file XML that supports this configuration, see [Job File Example for Configurations 1, 2, and 3](#) on page 3-19.

### Configuration 2: Copying Log Files to the Central Server Via a Script

In this configuration, the log files are written to a log folder on the server running the CIMS Windows Process collector. A file transfer script is then called by the collector at each logging interval to copy the log files on the collector server to a log folder on the central CIMS Data Collectors server. The log files on the central server are then processed into CSR files.

**Note** • The CIMS Server installation does not include a script for transferring files. If you need assistance developing a script, contact CIMS Lab. For information about using environment variables with scripts, see [page 3-6](#).

You can set up the file transfer script to run automatically using the **Run this command at each interval box** in the CIMS Windows Process Administrator GUI (see [page 3-13](#)).

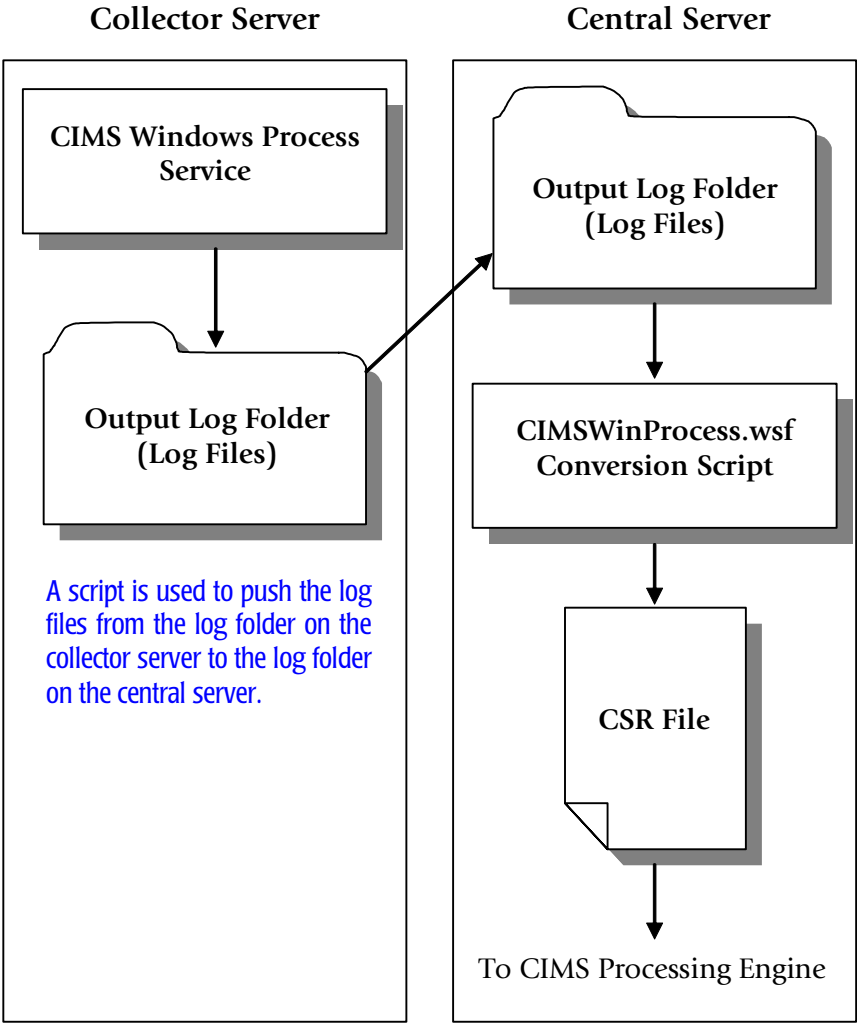


Figure 3-2 • System Configuration 2

For an example of the job file XML that supports this configuration, see [Job File Example for Configurations 1, 2, and 3](#) on page 3-19.

#### Using Environment Variables With the File Transfer Script

The CIMS Windows Process collector supports the following environment variables in addition to the standard environment variables provided with the Windows operating system (e.g., %COMPUTERNAME%):

- %CIMSDATE%      The date that the run command was issued.
- %CIMSTIME%      The time that the run command was issued.

You can use environment variables with the script in either of the following ways:

- Pass the variable from the command line. For example `C:\CopyLog.bat %CIMSDATE% %COMPUTERNAME%`. The CIMS Windows Process collector will expand the environment variables before launching the script.
- Use the `WshShell` object to enable the script to get the variable values directly from the environment. For more information, go to <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsproenvironment.asp>.

**Configuration 3: Writing Log Files Directly to the Central Server**

In this configuration, the log files are written directly to a log folder on the central CIMS Data Collectors server for processing.

**Note** • A disadvantage of this configuration is that if the network connection between the collector server and the central server is down, the log files are lost.

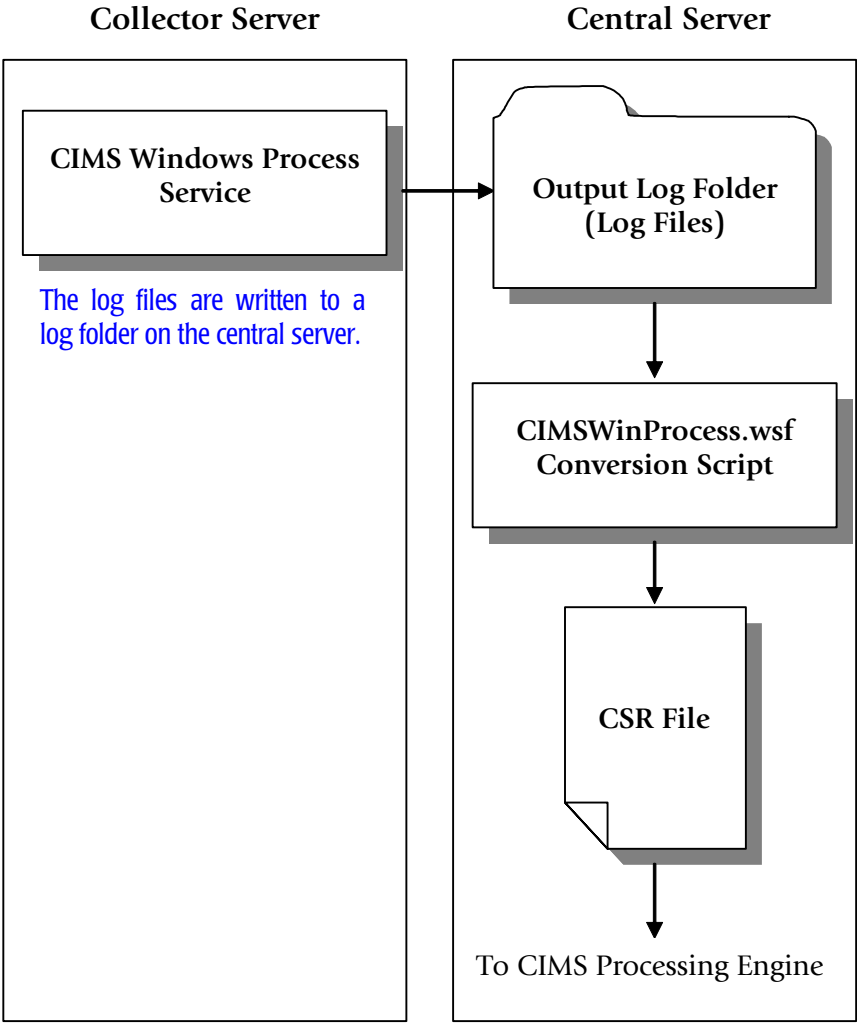


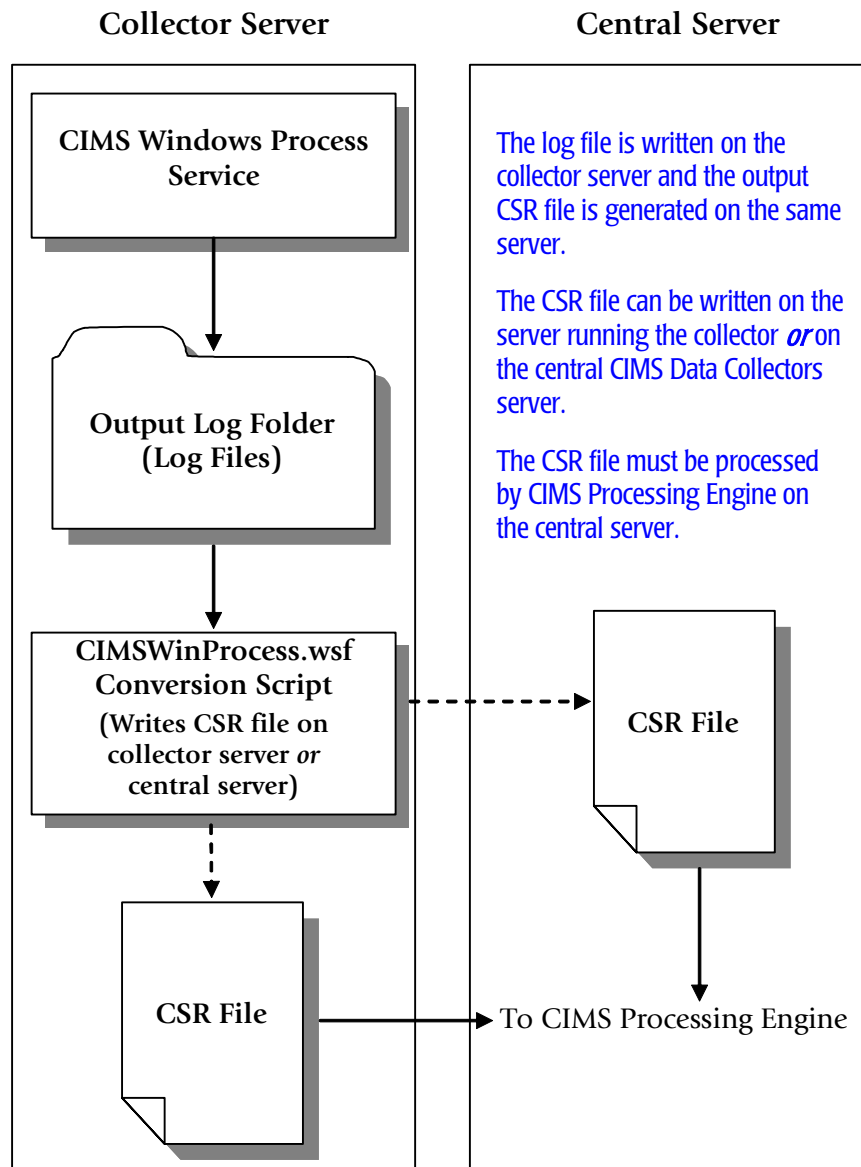
Figure 3-3 • System Configuration 3

For an example of the job file XML that supports this configuration, see *Job File Example for Configurations 1, 2, and 3* on page 3-19.

**Configuration 4: Generating CSR Files on the Collector Server**

**Note** • This configuration is usually not recommended. For more information, contact CIMS Lab.

In this configuration, the log files are written to a log folder on the server running the CIMS Windows Process collector. The CIMSWinProcess.wsf script is also run on this server. The output CSR records can be written on the server running the collector *or* on the central CIMS Data Collectors server; however, the CSR files must be processed by CIMS Processing Engine on the central server.



**Figure 3-4 • System Configuration 4**

For an example of the job file XML that supports this configuration, see *Job File Examples for Configuration 4* on page 3-22.

## Installing the CIMS Windows Process Collector

To use the CIMS Windows Process collector, you must have the CIMS Windows Process collector installed on the central CIMS Data Collectors server as described on [page 2-2](#).

In addition to installation on the central server, you need to install the CIMS Windows Process collector on each computer that you want to collect process data from. (In most cases, you will want to collect data for computers other than the central server.)

A simple setup program for the collector (CIMSWinProcessSetup.exe) is available in the following locations:

- On the Customer Downloads : CIMS Server page of the CIMS Lab Web site.
- In the CIMS\Server\CIMSWinProcess Install folder on the CIMS product CD.
- In the Collectors\CIMSWinProcess folder on the central CIMS Data Collectors server.

### *To install the CIMS Windows Process collector:*

---

**Note** • These following steps are also applicable if you are upgrading to a new version or release of the CIMS Windows Process collector.

---

- 1 Log on to Windows as an Administrator.
- 2 Click the Windows **Start** button, and then click **Run**.
- 3 Enter the path to the setup program CIMSWinProcessSetup.exe and then click **OK**.

The setup wizard appears.

- 4 Click **Next**.
- 5 Choose the default location for installation (C:\Program Files\CIMSLab) or click **Change** to choose another location. After making your selection, click **Next**.
- 6 In most cases, you will accept the default install features (i.e., leave the **Process Collector—Service** check box selected and all other check boxes unselected). The service collector is the most commonly used process collector. Contact CIMS Lab before selecting the **Process Collector—User32 Collector** check box.

Although it is not common, you might want to generate CSR files on the server running CIMS Windows Process collector as shown in [Configuration 4: Generating CSR Files on the Collector Server](#) on page 3-8. In this case, you also need to select the **Aggregation Engine** check box to install CIMS Aggregation Engine (CIMS\Aggregation.dll).

- 7 Click **Next**, and then click **Install**.
- 8 If the CIMS Windows Process service is run under a the Local System account (the default), leave the **User**, **Password**, and **Confirm** boxes blank and click **Next**.

If the account is other than Local System (see [page 3-2](#)), enter the user account, password, and password confirmation for the service, and then click **Next**.

- 9 Click **Finish** to complete the installation.

### Components Installed by CIMSWinProcessSetup.exe

The CIMSWinProcessSetup.exe setup program installs the following components:

- **The CIMS Windows Process service.** This is a Windows service that supports the collector. To view this service, in Windows Control Panel, open **Administrative Tools ▶ Services**.
- **The CIMS Windows Process collector.** This installs the following components in the Collectors\CIMSWinProcess folder created during installation:
  - The executable program for the collector, CIMSWinPService.exe.
  - An executable program, CIMSWinPServiceLog.exe, that is used by CIMS Lab for troubleshooting purposes. For more information about this program, contact CIMS Lab.
  - The executable program for the CIMS Windows Process collector's administrative program, CIMSWinPServiceAdmin.exe.
  - The conversion script, CIMSWinProcess.wsf. In most cases, this file is used on the central CIMS Data Collectors server and is not needed on other computers. The exception is if you are converting log files to CSR files on the computer running the CIMS Windows Process collector as shown in the configuration on [page 3-8](#).

If you selected the **Aggregation Engine** check box in the setup wizard, CIMS Aggregation Engine is also installed.

The installation does not include CIMS Processing Engine, which processes the CSR files created by CIMS Aggregation Engine and loads the output data into the database. To process CSR files, you need to process the files on the central CIMS Data Collectors server.



## Enabling CIMS Windows Process Logging

The CIMS Windows Process collector runs at configurable intervals and tracks all processes that are running at that time until the completion of the process. The usage data for each process is entered as a record or records in the log file.

The CIMS Windows Process collector includes an easy-to-use GUI administrative program for configuring and enabling the collection process. To use this program, click the **Start** menu, and then click **Programs** ▶ **CIMS Server** ▶ **Collectors** ▶ **CIMS Windows Process Administrator—Service** and set the following options:

### ■ Log File Output

- **Log file path.** Enter the path to the folder that you want to store the process log files in. If the file does not exist, you will be asked if you want to create the path. Click **Yes**.

The log file folder can be on the computer running the CIMS Windows Process collector or on the central CIMS Data Collectors server, depending on the system configuration that you are using for collection and processing (see [System Configuration Options for the CIMS Windows Process Collector](#) on page 3-4). You should create this folder in a location where you keep data that is backed up.

---

**Important!** • Do not set the log file path to the `Processes\CIMSWinProcess\<feed>` folder. The feed folder should contain only CSR files.

---

The default path is `C:\Program Files\CIMSLab\CIMSWinProcessLogs` (if you installed the CIMS Windows Process collector in the default location). The use of a UNC path for the log file location is recommended.

- **Log file prefix.** The default name for the log file is `CIMSProcessLog-yyyymmdd.txt`. You can use the default prefix `CIMSProcessLog-` or replace it with the prefix of your choice (or no prefix).
- **Use Local Time in output records.** If this check box is selected (the default), the local time set for the computer is used in the date and time fields in the log file. If this check box is cleared, Universal Time Coordinate (*UTC*) time is used in the log file.

---

**Note** • The date in the log file name always reflects local time, regardless of whether Use Local Time is selected.

---

#### ■ Sampling

- **Look for new process every.** Enter the number of seconds, minutes, or hours that you want to begin tracking new processes. For example, if you set the sampling interval to 5 seconds, the collector checks every 5 seconds to determine which new processes have begun since the last check and tracks those processes until completion.

You can use the sampling option alone or in conjunction with the interval accounting option. If you select the **Enable Interval Accounting** check box, a start, interval, and optional end record are created in the log file. If you do not select the **Enable Interval Accounting** check box, a cumulative End record is created in the log file when the process ends. (For a description of start, interval, and end records, see *CIMS Windows Process Collector Log File Format* on page 3-13.)

---

**Note** • The CIMS Windows Process collector does not collect data for processes that run between sampling intervals.

---

#### ■ Accounting

- **Enable Interval Accounting.** Select this check box to use interval accounting.

The use of interval accounting is recommended for chargeback because it provides Start, Interval, and optional End records for a process rather than just a cumulative End record. This is especially beneficial for long running processes that begin in one billing period and end in another.

When you select interval accounting, a Start record is created in the log file when the CIMS Windows Process collector begins tracking the process. Interval records are created at the interval times that you set in the **Write accounting records every** boxes or the **Write accounting records at** box until the process ends. If you select the **Write End records** check box, an End record containing a cumulative total for the process is also created.

- **Write End records.** Select this check box if you want End records to be included in the log file in addition to Start and Interval records. Because the End record provides cumulative totals of the usage totals shown in the Start and Interval records, you might not want to include End records when using interval accounting. For chargeback purposes, the resulting total usage amounts from the combined Start, Interval, and End records will be double the actual usage amount if the amounts are not filtered by the `CIMSWinProcess.wsf` script. For more information, contact CIMS Lab.

- **Write accounting records every.** This option enables you to create interval records at a set number of seconds, minutes, or hours from when the CIMS Windows Process collector was started.

For example, if you set this option to 15 minutes, an initial start record will be created for each process being tracked and subsequent interval records will be created every 15 minutes until the process ends. If you set this option to 24 hours, an interval record will be created every 24 hours for each process that is running until the process ends.

If you want to create interval records for all processes running at a specified time of day, use the **Write accounting records at** option.

- **Write accounting records at.** This option enables you to set a time each day (in 24 hour format) to produce interval records. An interval record is created for each process running at this time. This option is intended to be used to track longer running processes such as SQL Server, IIS, and other services. For these types of processes, you might want to create one daily interval record.
- **Run this command at each interval.** You can use this box to enter a command (for example, to run a .bat file or an executable) that will be executed at each logging interval. For an example of the use of this feature, see [Configuration 2: Copying Log Files to the Central Server Via a Script](#) on page 3-5.

- **Control Service.** Click this button to open the Service Control dialog box to start or stop the CIMS Windows Process service. You can also start and stop the service from Windows Control Panel and then click **Refresh Status** in the Service Control dialog box to make the change in the collector.

## CIMS Windows Process Collector Log File Format

The following table describes the record fields in the log file produced by the CIMS Windows Process collector.

There are three types of records that might appear in the log file:

- Start records, which provide usage data for the start of a process. The elapsed time in a Start record shows the amount of time in seconds that the process had been running when the collector began to track it. For example, if the process had been running for 2 minutes, the elapsed time for the Start record is 120.
- Interval records, which provide individual process usage data at each logging interval. The elapsed time in an Interval record is in seconds. For example, if interval accounting is set to 15 minutes, 900 seconds appear for each 15 minute interval that occurs while the process is running.
- If the process ends before the interval accounting begins, an interval record is created showing the time that the interval ran. Likewise, if the process ends between intervals, a final interval record is created showing the time that the interval ran.
- End records, which provide summary usage data at the end of a process. All totals in an End record are cumulative for the whole process.

Start and Interval records appear only if the collector is configured for interval accounting.

End records appear in the following situations:

- If the collector *is not* configured for interval accounting. In this situation, only End records appear.
- If the **Write End records** check box is selected for interval accounting.

---

**Note** • The term “process” in the following table can refer to the entire process, or the start, interval, or end of a process depending on whether interval accounting is used (see [page 3-12](#)).

---

Field Name	Description/Values
Record Type	S = Start of process (note that this does not appear if interval accounting is not used).  I = Interval (note that this does not appear if interval accounting is not used).  E = End of process (this record appears if you do not enable interval accounting or if you enable interval accounting and select the <b>Write End records</b> check box).
ProcessID	Process identifier (PID) assigned to the process by the operating system.
ParentProcessID	The PID for the entity that created the process. Assigned by the operating system.
ProcessName	The name of the process.
ProcessPath	The path where the process executable is located.
MachineName	The name of the computer running the process.
UserName	The name of the user that created the process.
TerminalServicesSessionID	If Microsoft Terminal Services is used to access the process on the computer, the session ID.
CreateDateTime	The date and time that the process was created.
ExitDateTime	The date and time that the entire process ended.
ExitCode	The result from the completion of the process.
IntervalStartDateTime	If using interval accounting, the date and time that the interval started.

**Table 3-1 • CIMS Windows Process Collector Log File Format**

Field Name	Description/Values
IntervalEndDateTime	If using interval accounting, the date and time that the interval ended.
ElapsedTimeSecs	The total elapsed time in seconds for the process.
CPUTimeSecs	The total elapsed CPU time in seconds for the process. This field is the sum of KernelCPUTimeSecs and the UserCPUTimeSecs fields.
KernelCPUTimeSecs	The total elapsed time in seconds that the process spent in kernel mode. (For a description of kernel mode, see <i>About Kernel Mode and User Mode</i> on page 3-17).
UserCPUTimeSecs	The total elapsed time in seconds that the process spent in user mode. (For a description of user mode, see <i>About Kernel Mode and User Mode</i> on page 3-17).
Read Requests	The number of read requests made by the process.
KBytesRead	The number of kilobytes read by the process.
Write Requests	The number of write requests made by the process.
KBytesWritten	The number of kilobytes written by the process.
PageFaultCount	In a paged virtual memory system, an access to a page (block) of memory that is not currently mapped to physical memory. When a page fault occurs, the operating system either fetches the page from secondary storage (usually disk) if the access is legitimate or reports the access as illegal if access is not legitimate. A large number of page faults lowers performance.
WorkingSetSizeKB	The amount of memory in kilobytes mapped into the process context.
PeakWorkingSetSizeKB	The maximum amount of memory in kilobytes mapped into the process context at a given time.
PagefileUsageKB	The amount of memory in kilobytes that is set aside in the system swapfile for the process. It represents how much memory has been committed by the process.
PeakPagefileUsageKB	The maximum amount of memory in kilobytes that is set aside in the system swapfile for the process.
PriorityClass	The priority class for the process. Assigned by the operating system.
BasePriority	The priority with which the process was created. Assigned by the operating system.

**Table 3-1 • CIMS Windows Process Collector Log File Format (Continued)**

## ■ Operating System Data Collectors

### *Windows Process Data Collector*

---

Field Name	Description/Values
SystemProcessorCount	The number of processors on the computer.
EligibileProcessorCount	The number processors on the computer that the process is allowed to use.
AffinityMask	A bit mask value indicating which processors the process may run on.

---

**Table 3-1 • CIMS Windows Process Collector Log File Format (Continued)**

## About Kernel Mode and User Mode

The kernel mode is where the computer operates with critical data structures, direct hardware (IN/OUT or memory mapped), direct memory, interrupt requests (IRQs), direct memory access (DMA), etc.

The user mode is where users can run applications. The kernel mode prevents the user mode from damaging the system and its features.

Figure 3-5 shows the relationship of the kernel and user mode.

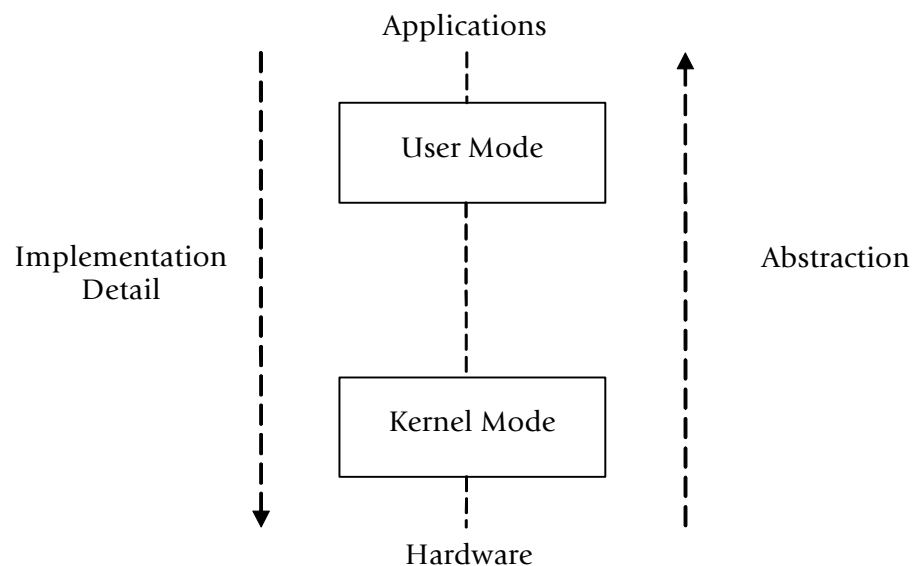


Figure 3-5 • Kernel and User Mode

## Identifiers and Resources Collected From the CIMS Windows Process Collector Log File

By default, the following fields in the CIMS Windows Process collector log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `CIMSWinProcess.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>		
—	Feed (defined in the CIMS Windows Process collector job file)	—
ProcessName	ProcessName	—
ProcessPath	ProcessPath	—
MachineName	Server	—
UserName	User	—
PriorityClass	PriorityClass	—
BasePriority	BasePriority	—
<b>Resources</b>		
ElapsedTimeSecs	MS Windows Elapsed Time	WINELPTM
CPUTimeSecs	MS Windows CPU Time	WINCPUTM
KernelCPUTimeSecs	MS Windows Kernel CPU Time	WINKCPUT
UserCPUTimeSecs	MS Windows User CPU Time	WINCPUUS
Read Requests	MS Windows Read Requests	WINRDREQ
KBytesRead	MS Windows KB Read	WINKBYTR
Write Requests	MS Windows Write Requests	WINWRREQ
KBytesWritten	MS Windows KB Written	WINKBWRI
PageFaultCount	MS Windows Page Fault Count	WINPGFLT

**Table 3-2 • Default CIMS Windows Process Identifiers and Resources**



## Setting Up the CIMS Windows Process Collector

You will need to set up an XML job file on the central CIMS Data Collectors server for all system configurations. If you are using the system configuration shown in [Configuration 4: Generating CSR Files on the Collector Server](#) on page 3-8, you also need to set up a job file on the server running the CIMS Windows Process collector.

The following sections provide job file examples by system configuration type.

### Job File Example for Configurations 1, 2, and 3

On the central CIMS Data Collectors server, set up an XML job file for the CIMS Windows Process collector as described in [Creating Job Files](#) on page 2-25. The following is an example process for the CIMS Windows Process collector in the job file. Note that the location of the log folder is defined by the `LogFolder` parameter. Depending on the system configuration that you are using, the value for this parameter will be the path to the log folder on the server running the collector (Configuration 1) or the path to the log folder on the central server (Configuration 2 or 3).

```
<Process      id="CIMSWinProcess"
             description="Process for CIMS Windows Process Collection"
             active="true">
  <Steps>
    <Step      id="Server1 Collection"
             description="Server1 CIMSWinProcess"
             type="ConvertToCSR"
             programName="CIMSWinProcess\CIMSWinProcess.wsf"
             programType="wsf"
             active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="//Server1\CIMSWinProcessLogs"/>
      </Parameters>
    </Step>
    <Step      id="Scan"
             description="Scan CIMSWinProcess"
             type="Process"
             programName="Scan"
             programType="net"
             active="true">
      </Step>
    <Step      id="Process"
             description="Standard Processing for CIMSWinProcess"
             type="Process"
             programName="SingleProcessStep"
             programType="com"
             active="true">
      </Step>
    <Step      id="DatabaseLoad"
             description="Database Load for CIMSWinProcess"
             type="Process"
             programName="DBLoad"
             programType="com"
             active="true">
      </Step>
    <Step      id="Cleanup"
             description="Cleanup CIMSWinProcess"
```

```

        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
    <Parameters>
        <Parameter DaysToRetainFiles="45"/>
    </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the CIMS Windows Process collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 3-3](#). These parameters are used by the conversion script, `CIMSWinProcess.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

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 <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the log file that you want to process. If the log file is on the same server as the <code>CIMSWinProcess.wsf</code> script used to convert the file, you can also use "Self" and the server name is defined automatically (see the example on <a href="#">page 3-22</a>).</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>

**Table 3-3 • CIMSWinProcess.wsf Parameters**

Parameter	Description/Values
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>By default, the output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="CIMSWinProcess"</code>, the output folder is <code>CIMSWinProcess</code>.</p> <p>This parameter is required only if you are running the <code>CIMSWinProcess.wsf</code> script on one server, but want to send CSR files to a process definition folder on another server. (This configuration is not common.) In this case, you need to provide the path to the process definition folder.</p>
LogFolder	The location of the log file to be processed.

---

**Table 3-3 • CIMSWinProcess.wsf Parameters (Continued)**

## Job File Examples for Configuration 4

The job file XML for this system configuration differs depending on whether CSR files are written to the server running the CIMS Windows Process collector or the central CIMS Data Collectors server.

### *To write the CSR files to the server running the CIMS Windows Process collector:*

On the computer running the CIMS Windows Process collector, set up a job file as described in [Creating Job Files](#) on page 2-25. The process for the collector in the job file should contain only a `ConvertToCSR` step and a `FileTransfer` step as shown in the following example.

In this example, the log files are written to the collector server (Server1). CSR files created from the log files will be copied from the `CIMSWinProcess\Server1` folder on the collector server to the `CIMSWinProcess\Server1` folder on the central server (CIMS).

The `%LogDate_End%` variable in the `FileTransfer` from parameter specifies that CSR files that contain a date matching the last day of the `LogDate` parameter are copied. For example, if the `LogDate` parameter is the default `PREDAY`, CSR files with the previous day's date are copied. (For more information about the `LogDate` parameter, see [Specifying Log Dates for Collection](#) on page 2-3.)

```
<Process id="CIMSWinProcess"
  description="Process for CIMS Windows Process Collection"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 CIMSWinProcess"
      type="ConvertToCSR"
      programName="CIMSWinProcess\CIMSWinProcess.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Self"/>
        <Parameter LogFolder="\\Server1\CIMSWinProcessLogs"/>
      </Parameters>
    </Step>
    <Step id="FileTransfer"
      description="Transfer CSR Files"
      type="Process"
      programName="FileTransfer"
      programType="net"
      active="true">
      <Parameters>
        <Parameter type="Windows"/>
        <Parameter from="\\Server1\CIMSWinProcess\Server1\
          %LogDate_End%.txt"
          to="\\CIMS\CIMSWinProcess\Server1"
          action="Copy"
          overwrite="true"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

On the central CIMS Data Collectors server, set up a job file that does not contain the ConvertToCSR step for the collector (i.e., Scan is the first step). For example:

```
<Process id="CIMSWinProcess"
  description="Process for CIMS Windows Process Collection"
  active="true">
  <Steps>
    <Step id="Scan"
      description="Scan CIMSWinProcess"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for CIMSWinProcess"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    :
    :
```

### **To write the CSR files to the central CIMS Data Collectors server:**

On the computer running the CIMS Windows Process collector, set up a job file as described in [Creating Job Files](#) on page 2-25. The process for the collector in the job file should contain only a ConvertToCSR step as shown in the following example.

In this example, the log files are written to the collector server (Server1). The path for the log file folder is specified by the LogFolder parameter.

The CSR files created from the log files are written to the central CIMS Data Collectors server (CIMS) using the path specified by the OutputFolder parameter and the value of the Feed parameter. That is, CSR files will be written to the feed subfolder Server1 in the CIMSWinProcess process definition folder on the central server.

```
<Process id="CIMSWinProcess"
  description="Process for CIMS Windows Process Collection"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 CIMSWinProcess"
      type="ConvertToCSR"
      programName="CIMSWinProcess\CIMSWinProcess.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\CIMSWinProcessLogs"/>
        <Parameter OutputFolder="\\CIMS\CIMSWinProcess"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

On the central CIMS Data Collectors server, set up a job file that does not contain the ConvertToCSR step for the collector (i.e., Scan is the first step). For example:

```
<Process id="CIMSWinProcess"
  description="Process for CIMS Windows Process Collection"
  active="true">
  <Steps>
    <Step id="Scan"
      description="Scan CIMSWinProcess"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for CIMSWinProcess"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
  :
  :
```

## Running the CIMS Windows Process Collector

Use CIMS Job Runner to run the CIMS Windows Process collector as described in [Running CIMS Job Runner](#) on page 2-93.

## Windows System Resource Manager (WSRM) Collector

The CIMS Data Collector for WSRM gathers usage data for processes running on the Windows Server 2003 Enterprise and Datacenter operating systems. This data is contained a log file produced by WSRM, which provides useful metrics such as:

- The name of the process.
- Name of the user that created the process.
- The elapsed CPU time used by the process (cumulative and broken down by kernel and user time).
- Bytes read and written by the process.

This collector uses the same components as the Universal collector as described in [Chapter 14, CIMS Universal Data Collector](#).

### Identifiers and Resources Collected by the WSRM Collector

By default, the following values in the log file are defined as chargeback identifiers and resource rate codes. (The rate codes assigned to the resources are pre-loaded in the CIMSRate table.) These identifiers and resources are defined in the conversion definition file `WSRMDef.txt`. This file contains the conversion information required by CIMS Conversion Engine to create a CSR file from the WSRM log file. If you installed CIMS Server in the default location, this file is in `C:\Program Files\CIMSLab\Collectors\WSRM`.

#### Identifiers

- ProcessID
- ProcessName
- Domain
- User
- ProgramPath

#### Resources

- WINELPTM (MS Windows Elapsed Time)
- WINCPUPTM (MS Windows CPU Time)
- WINKCPUT (MS Windows Kernel CPU Time)
- WINCPUUS (MS Windows User CPU Time)
- WINRDREQ (MS Windows Read Requests)
- WINKBYTR (MS Windows KB Read)
- WINWRREQ (MS Windows Write Requests)
- WINKBWRI (MS Windows KB Written)

## Setting Up the WSRM Collector

On the central CIMS Data Collectors server, set up an XML job file for the WSRM collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file. Because this collector uses CIMS Conversion Engine, the conversion script for the Universal collector (Universal.wsf) is called.

```
<Process id="WSRM"
description="Process for WSRM"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 WSRM"
type="ConvertToCSR"
programName="Universal\Universal.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter ConvEngDefName="C:\Program Files\CIMSLab\Collectors\
WSRM\WSRMDef.txt"/>
        <Parameter InputFileName="C:\WSRMLogs\WSRMAccountingInfo.csv"/>
        <Parameter OutputFolder="%ProcessFolder%"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan WSRM"
type="Process"
programName="Scan"
programType="net"
active="true">
    </Step>
    <Step id="Process"
description="Standard Processing for WSRM"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
    <Step id="DatabaseLoad"
description="Database Load for WSRM"
type="Process"
programName="DBLoad"
programType="com"
active="true">
    </Step>
    <Step id="Cleanup"
description="Cleanup WSRM"
type="Process"
programName="Cleanup"
programType="net"
active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```



For a description of the `Parameter` element attributes that are specific to the Universal collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 3-4](#). These parameters are used by the conversion script, `Universal.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The log date specifies the date that appears in the initial CSR file name. This is the CSR file that is processed by the Scan program. The start and end dates that appear in the CSR file records are defined by the definition file <code>WSRMDef.txt</code>.</p> <p>For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="WSRM"</code>, the output folder is <code>WSRM</code>.</p>
ConvEngDefName	The location of the conversion definition file <code>WSRMDef.txt</code> .
InputFileName	The location of the log file to be processed.

**Table 3-4 • Universal.wsf Parameters**

# Citrix Data Collector

The CIMS Data Collector for Citrix collects data that is contained in the Citrix Resource Manager summary database. For more information about this database, refer to the Citrix documentation at <http://support.citrix.com/docs/>.

The Citrix collector provides CPU time and memory used by user, server, and process.

## Identifiers and Resources Collected by the Citrix Collector

By default, the following data collected by the Citrix collector is defined as chargeback identifiers and resource rate codes (see the `AddIdentifier` and `AddResource` methods in the `Citrix.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

### Identifiers

- Feed (defined in the Citrix collector job file)
- UserName (the user that accessed the application or information)
- ServerName (the Citrix server from which the application/information was accessed)
- ProcessName (the process started by the user in the Citrix session)

### Resources

- CTRXCPU (CPU time used)
- CTRXMEM (Memory used)

## Setting Up the Citrix Collector

### Create a CIMS Data Source

You need to create a CIMS Data Source that points to the Citrix Resource Manager summary database. For the steps required to create a CIMS Data Source, see [Appendix B](#).

### Set Up the Job File

On the central CIMS Data Collectors server, set up an XML job file for the Citrix collector as described in [Creating Job Files](#) on page 2-25. The following is an example process for the collector in the job file.

```
<Process id="Citrix"
  description="Process for Citrix Collection"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 Citrix"
      type="ConvertToCSR"
      programName="Citrix\Citrix.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter DataSourceID="CitrixDB"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan Citrix"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for Citrix"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for Citrix"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup Citrix"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
```

```

        <Parameters>
          <Parameter DaysToRetainFiles="45"/>
        </Parameters>
      </Step>
    </Steps>
  </Process>

```

For a description of the `Parameter` element attributes that are specific to the Citrix collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 3-3](#) on page 3-20. These parameters are used by the conversion script, `Citrix.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

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 <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the Citrix Resource Manager summary database that you want to collect data from.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="Citrix"</code>, the output folder is <code>Citrix</code>.</p>
DataSourceID	<p>The CIMS Data Source ID for the Resource Manager summary database.</p> <p>Do not leave this parameter blank. If this parameter is blank, the CIMS Data Source that is set as the Web/Collector default in CIMS Server Administrator is used. The default CIMS Data Source should point to the CIMS Server database.</p>
owner (optional)	The owner of the tables in the Citrix database if other than <code>dbo</code> . Make sure that you include the period after the owner's name (e.g., <code>John.</code> ).

**Table 3-5 • Citrix.wsf Parameters**

## Running the Citrix Collector

Use CIMS Job Runner to run the Citrix collector as described in *Running CIMS Job Runner* on page 2-93.

## VMware Data Collector

The CIMS Data Collector for VMware collects data that is contained in the VMware VirtualCenter database. For more information about this database, refer to the VMware documentation at <http://www.vmware.com/support/pubs/>.

The VMware collector provides CPU, disk, memory, and network usage by user host, virtual machine, and virtual machine group.

## Identifiers and Resources Collected by the VMware Collector

By default, the following data collected by the VMware collector is defined as chargeback identifiers and resource rate codes (see the `AddIdentifier` and `AddResource` methods in the `VMware.wsf` conversion script). The rate codes assigned to the resource *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

### Identifiers

- Feed (defined in the VMware collector job file)
- VMName (the name of the virtual machine)
- ConfigFileName (path of the virtual machine configuration file)
- UserName (the user name to connect to the host)
- HostName (the ID of the host)
- VMGroupName (the ID of the virtual machine group)

### Resource

- VMCPUUSE (CPU usage)
- VMCPUGUA (CPU usage guaranteed)
- VMDSKRED (disk kilobytes read)
- VMDSKWRI (disk kilobytes written)
- VMMEMACT (memory kilobytes active)
- VMMEMGRT (memory kilobytes granted)
- VMNETREC (network kilobytes read)
- VMNETTRN (network kilobytes transferred)

## Setting Up the VMware Collector

### Create a CIMS Data Source

You need to create a CIMS Data Source that points to the VMware VirtualCenter database. For the steps required to create a CIMS Data Source, see [Appendix B](#).

### Set Up the Job File

On the central CIMS Data Collectors server, set up an XML job file for the VMware collector as described in [Creating Job Files](#) on page 2-25. The following is an example process for the collector in the job file.

```
<Process id="VMware"
description="Process for VMware Collection"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 VMware"
type="ConvertToCSR"
programName="VMware\VMware.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter DataSourceID="VMWDB"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan VMware"
type="Process"
programName="Scan"
programType="net"
active="true">
    </Step>
    <Step id="Process"
description="Standard Processing for VMware"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
    <Step id="DatabaseLoad"
description="Database Load for VMware"
type="Process"
programName="DBLoad"
programType="com"
active="true">
    </Step>
    <Step id="Cleanup"
description="Cleanup VMware"
type="Process"
programName="Cleanup"
programType="net"
active="true">
```

```

        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the VMware collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 3-3](#) on page 3-20. These parameters are used by the conversion script, `VMware.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

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 <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the VMware database that you want to collect data from.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id</code>="VMware", the output folder is <code>VMware</code>.</p>

**Table 3-6 • VMware.wsf Parameters**

Parameter	Description/Values
MSAccessDBLocation (optional)	The full path to the VMware VirtualCenter Microsoft Access database (.MDF file).  If you are using a SQL Server or Oracle database, use the DataSourceID parameter.
DataSourceID (optional)	The CIMS Data Source ID for the VMware VirtualCenter SQL Server or Oracle database. If you are using a Microsoft Access database, use the MSAccessDBLocation parameter.  If you use this parameter, do not leave the value blank. If this parameter is blank, the CIMS Data Source that is set as the Web/Collector default in CIMS Server Administrator is used. The default CIMS Data Source should point to the CIMS Server database.

---

**Table 3-6 • VMware.wsf Parameters (Continued)**

## Running the VMware Collector

Use CIMS Job Runner to run the VMware collector as described in *Running CIMS Job Runner* on page 2-93.

## AS/400 Data Collector

CIMS Lab provides a CIMS Data Collector for AS/400. For instructions on how to configure this collector, contact CIMS Lab (*Chapter 15, Contacting Technical Support*).



# Database Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for databases. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Microsoft SQL Server 2000 Data Collector</b> .....	<b>4-3</b>
Enabling SQL Server 2000 Tracing .....	4-4
SQL Server 2000 Trace File Format .....	4-6
Identifiers and Resources Collected from the SQL Server 2000 Trace File .....	4-8
Setting Up the SQL Server 2000 Collector .....	4-9
Running the SQL Server 2000 Collector .....	4-17
<b>Oracle Data Collector</b> .....	<b>4-18</b>
Setting Up the CIMSWIND Collector .....	4-18
Creating a Process Definition Folder for Oracle Data Collection .....	4-21
Enabling Oracle Logging .....	4-21
Resources Collected .....	4-23
Setting Up the Oracle Collector Job File .....	4-24
Running the Oracle Collector .....	4-25
<b>DB2 Data Collector</b> .....	<b>4-26</b>
Setting up the CIMSWIND Collector .....	4-26
Creating a Process Definition Folder for DB2 Data Collection .....	4-26
Enabling DB2 Logging .....	4-27
Resources Collected .....	4-29
Setting Up the DB2 Collector Job File .....	4-31
Running the DB2 Collector .....	4-32
<b>Sybase Data Collector</b> .....	<b>4-33</b>

<b>Database Size Data Collector (DBSpace)</b> .....	<b>4-33</b>
Identifiers and Resources Collected by the DBSpace Collector .....	4-33
Setting Up the DBSpace Collector .....	4-33
Running the DBSpace Collector .....	4-36
<b>SQL Server Reporting Services Data Collector</b> .....	<b>4-37</b>
Identifiers and Resources Collected by the Reporting Services Collector .....	4-37
Setting Up the Reporting Services Collector .....	4-38

## Microsoft SQL Server 2000 Data Collector

The CIMS Data Collector for Microsoft SQL Server 2000 collects data that is contained in a trace file produced by SQL Server. This trace provides useful metrics such as the following for each database in a SQL Server instance:

- SQL Server login name or Windows NT user name.
- Amount of elapsed time taken by an event.
- Amount of CPU time used by an event.
- Number of logical disk reads performed by the server on behalf of the event.
- Number of physical disk writes performed by the server on behalf of the event.

The CIMS Lab stored procedure `CIMSSp_SQLServer2000Trace` calls Microsoft stored procedures to create the trace file. Instructions for installing and running this stored procedure are provided in *Enabling SQL Server 2000 Tracing* on page 4-4.

The following sections provide steps for enabling tracing for SQL Server 2000 and for setting up and running the SQL Server 2000 collector.

---

**Note** • The SQL Server 2000 collector supports SQL Server clusters. A cluster refers to a group of two or more servers that work together and represent themselves as a single virtual server to a network.

---

### Creating a Trace File Folder for Storing Trace Files

Before you run the `CIMSSp_SQLServer2000Trace` stored procedure, you need to create a trace file folder for storing the trace files. The trace file folder should be on the database server that you are collecting data from. (It is possible, but not recommended, that the folder can be on another server.)

You should create the trace file folder in a location where you keep data that is backed up. In addition, make sure that the Windows account that you are running the SQL Server data collector under has permission to write to the trace file folder.

The location of the trace file folder is defined as a parameter in the job file for the SQL Server 2000 collector job. CIMS Job Runner passes this parameter to the conversion script. See *Setting Up the SQL Server 2000 Collector* on page 4-9.

## Enabling SQL Server 2000 Tracing

The CIMSSp\_SQLServer2000Trace stored procedure performs the following functions:

- Stops the current SQL trace logging.
- Closes the current trace file in the trace file folder.
- Starts a new trace file in the trace file folder.

Once installed, the stored procedure should be scheduled to run once a day (see *Running the CIMSSp\_SQLServer2000Trace Stored Procedure* on page 4-5).

CIMSSp\_SQLServer2000Trace uses the following built-in Microsoft SQL Server 2000 stored procedures. For more information about these stored procedures, use the link to go to the description on the Microsoft Web site.

- sp\_trace\_setstatus

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts\\_sp\\_ta-tz\\_0tnt.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_sp_ta-tz_0tnt.asp)

- sp\_trace\_create

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts\\_sp\\_ta-tz\\_8h49.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_sp_ta-tz_8h49.asp)

- sp\_trace\_setevent

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts\\_sp\\_ta-tz\\_1c4p.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_sp_ta-tz_1c4p.asp)

### Installing the CIMSSp\_SQLServer2000Trace Stored Procedure

To create SQL Server trace files, you need to install the CIMSSp\_SQLServer2000Trace stored procedure in one database on each SQL Server instance that you want to collect data from. You can install the stored procedure in the master database or any other database in the instance.

---

**Note** • If you run the stored procedure from a job file, the CIMS Data Source ID for the database that contains the stored procedure must be entered as a parameter for the MSSQL2000.wsf script (see page 4-10 for an example).

---

CIMS Lab provides a script, InstallSQLTrace.bat, that you can use to install the stored procedure. (If you installed CIMS Data Collectors in the default location, this script is in C:\Program Files\CIMSLab\Collectors\MSSQLServer\2000.) To use InstallSQLTrace.bat:

- 1 In the script file, change the -d parameter CIMSServer to your database name.
- 2 Edit the other parameters as needed. For example, change the -i parameter if the stored procedure SQLServer2000Trace.sql (which creates CIMSSp\_SQLServer2000Trace) is stored in another location.
- 3 Run the script to place the CIMSSp\_SQLServer2000Trace stored procedure in the database.

If you have databases on multiple servers, you need to change the parameters as needed and run the script for each server.

## Running the CIMSSp\_SQLServer2000Trace Stored Procedure

**Note** • Make sure that you have created a folder for storing trace files before running the CIMSSp\_SQLServer2000Trace stored procedure (see [Creating a Trace File Folder for Storing Trace Files](#) on page 4-3).

You can run the CIMSSp\_SQLServer2000Trace stored procedure from a job file or you can use SQL Server scheduling tools to run the stored procedure. Review the following sections to determine the method that you should use.

### Running the Stored Procedure From a Job File

This method consolidates the creation and processing of the trace file in one location. However, this method requires that the ID that you are running CIMS Job Runner with has `sysadmin` authority on the production SQL Server machine. The ID must be a member of the `sysadmin` server role to run the CIMSSp\_SQLServer2000Trace stored procedure.

Running the stored procedure from a job file uses the schedule that you have set up in Windows Task Scheduler for CIMS Job Runner. No additional scheduling is required. The stored procedure is run as part of the collection process and the resulting trace file is placed in the trace file folder. (The path to the trace file folder is defined in the job file.) The trace file is then pulled from the trace file folder for processing.

### Running the Stored Procedure Using SQL Server Scheduling Tools

This method is typically used because the ID that you are running CIMS Job Runner with does not have `sysadmin` authority on the production SQL Server machine. In this case, you cannot run the CIMSSp\_SQLServer2000Trace stored procedure from a job file. The stored procedure must be run from the production server using an ID that is a member of the `sysadmin` server role.

If the stored procedure is run using SQL Server scheduling tools, you need to provide a path to the trace file folder as a parameter when you schedule the SQL Server job. The trace file will then be placed in this path and pulled from the trace file folder for processing.

## Modifying the SQLServer2000Trace Stored Procedure (Optional)

**Note** • Modifying the SQLServer2000Trace stored procedure is not recommended.

The SQLServer2000Trace.sql stored procedure creates the CIMSSp\_SQLServer2000Trace stored procedure that is installed on the database server(s). (If you installed CIMS Data Collectors in the default location, SQLServer2000Trace.sql is in C:\Program Files\CIMSLab\Collectors\MSSQLServer\2000.)

SQLServer2000Trace.sql defines the event that causes data to be logged to the trace file and the event columns (data) that appear in the trace file (see *SQL Server 2000 Trace File Format* on page 4-6). You can change the event and event columns in the stored procedure. However, the columns are defined as identifiers and resources in the MSSQL2000.wsf conversion script. If you change the columns, you need to modify the script. You also need to add any new rate codes to the CIMSRate table as described in the *CIMS Server Administrator's Guide*.

## SQL Server 2000 Trace File Format

Data is logged to the trace file when the SQL Server event ID 15, Logout (the user logs out of SQL Server), occurs.

The following table describes the event columns that are provided in the trace file.

Column Name	Description/Values
TextData	Text value dependent on the event class that is captured in the trace.
BinaryData	Binary value dependent on the event class captured in the trace.
DatabaseID	ID of the database specified by the USE <database> statement, or the default database if no USE <database> statement is issued for a given connection.  The value for a database can be determined by using the DB_ID function.
TransactionID	System-assigned ID of the transaction.
Reserved	
NTUserName	Microsoft Windows NT user name.
NTDomainName	Windows NT domain to which the user belongs.
ClientHostName	Name of the client computer that originated the request.
ClientProcessID	ID assigned by the client computer to the process in which the client application is running.

**Table 4-1 • SQL Server 2000 Trace File Format**

Column Name	Description/Values
ApplicationName	Name of the client application that created the connection to an instance of SQL Server. This column is populated with the values passed by the application rather than the displayed name of the program.
SQLSecurityLoginName	SQL Server login name of the client.
SPID	Server Process ID assigned by SQL Server to the process associated with the client.
Duration	Amount of elapsed time (in milliseconds) taken by the event. This data column is not populated by the Hash Warning event.
StartTime	Time that the event started, when available.
EndTime	Time that the event ended. This column is not populated for starting event classes, such as SQL:BatchStarting or SP:Starting. It is also not populated by the Hash Warning event.
Reads	Number of logical disk reads performed by the server on behalf of the event. This column is not populated by the Lock:Released event.
Writes	Number of physical disk writes performed by the server on behalf of the event.
CPU	Amount of CPU time (in milliseconds) used by the event.
ObjectName	Name of object accessed.
DatabaseName	Name of the database specified in the USE <database> statement.
Filename	Logical name of the file name modified.
ObjectOwner	Owner ID of the object referenced.
TargetRoleName	Name of the database or server-wide role targeted by a statement.
TargetUserName	User name of the target of some action.
DatabaseUserName	SQL Server database user name of the client.
ServerName	Name of the instance of SQL Server (either servername or servername\instancename) being traced.

---

**Table 4-1 • SQL Server 2000 Trace File Format (Continued)**

## Identifiers and Resources Collected from the SQL Server 2000 Trace File

By default, the following fields in the SQL Server 2000 trace file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `MSSQL2000.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

Trace File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>		
—	Feed (defined in the SQL Server 2000 collector job file)	—
BinaryData	EventClass	—
SPID	SPID	—
SQLSecurityLoginName	LoginName	—
ApplicationName	ApplicationName	—
TextData	TextData	—
NTDomainName	NTDomainName	—
NTUserName	User	—
ClientHostName	HostName	—
ClientProcessID	ClientProcessID	—
ServerName	Server	—
DatabaseName	DatabaseName	—
DatabaseID	DatabaseID	—
<b>Resources</b>		
—	MS Windows SQL Server Records This is the number of records in the log file. That is, each time an event ID 15 occurs, a record is added to the log file. This resource is passed from the <code>MSSQL2000.wsf</code> conversion script.	SQLREC
Duration	MS Windows SQL Server Duration	SQLDUR
CPU	MS Windows SQL Server CPU	SQLCPU

**Table 4-2 • Default SQL Server 2000 Identifiers and Resources**



Trace File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
Reads	MS Windows SQL Server Reads	SQLREADS
Writes	MS Windows SQL Server Writes	SQLWRITE

Table 4-2 • Default SQL Server 2000 Identifiers and Resources (Continued)

## Setting Up the SQL Server 2000 Collector

On the central CIMS Data Collectors server, set up an XML job file for the SQL Server 2000 collector as described in *Creating Job Files* on page 2-25. The following are three example processes for the SQL Server collector in the job file:

- *Job File Example 1: Run Stored Procedure From a Job File* on page 4-10 supports the scenario in which the CIMSSp\_SQLServer2000Trace stored procedure is run from the job file, the trace file is placed in the trace file folder on the database server, and the trace file is then pulled from the folder for processing.
- *Job File Example 2: Run Stored Procedures Using SQL Server Tools and Pull Files to CIMS Data Collectors Server* on page 4-12 supports the scenario in which the stored procedure is run using SQL Server tools, the trace file is placed in the trace file folder on the database server, and the trace file is then pulled from the folder for processing.
- *Job File Example 3: Run Stored Procedures Using SQL Server Tools and Transfer Files to CIMS Data Collectors Server* on page 4-14 supports a scenario in which the stored procedure is run using SQL Server tools, but the trace file is not pulled from the trace file folder on the database server. Instead, a copy of the file is transferred to a trace file folder on the central CIMS Data Collectors server for processing. This method provides the greatest level of security because CIMS Job Runner does not need to access the database server other than to transfer the trace file.

In each of these examples, the database server is Server2 and the CIMS Data Collectors server is Server1.

### Job File Example 1: Run Stored Procedure From a Job File

In this example, the parameter `RunSP="true"` specifies that the `MSSQL2000.wsf` script is used to run the `CIMSSp_SQLServer2000Trace` stored procedure. The parameter `DataSourceID="DB1"` specifies the CIMS Data Source for the database that contains the stored procedure.

The trace file folder on the database server is in a shared folder named `SQLTraceFolder`. The parameter `TraceFolder="\\Server2\SQLTraceFolder"` specifies that the trace file will be placed in this shared folder.

```
<Process id="MSSQL2000"
description="Process for SQL Server 2000 Collection"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 MSSQL2000"
type="ConvertToCSR"
programName="MSSQLServer\2000\MSSQL2000.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server2"/>
        <Parameter TraceFolder="\\Server2\SQLTraceFolder"/>
        <Parameter DataSourceID="DB1"/>
        <Parameter RunSP="true"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan MSSQL2000"
type="Process"
programName="Scan"
programType="net"
active="true">
  </Step>
    <Step id="Process"
description="Standard Processing for MSSQL2000"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
  </Step>
    <Step id="DatabaseLoad"
description="Database Load for MSSQL2000"
type="Process"
programName="DBLoad"
programType="com"
active="true">
  </Step>
    <Step id="Cleanup"
description="Cleanup MSSQL2000"
type="Process"
programName="Cleanup"
programType="net"
active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
  </Step>
  </Steps>
</Process>
```

```
    </Step>  
  </Steps>  
</Process>
```

For a description of each of the parameters for the MSDDL2000.wsf script, see [Table 4-3](#) on page 4-16. For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

## Job File Example 2: Run Stored Procedures Using SQL Server Tools and Pull Files to CIMS Data Collectors Server

In this example, the parameter `RunSP="false"` specifies that the `MSSQL2000.wsf` script is not used to run the `CIMSSp_SQLServer2000Trace` stored procedure. This example assumes that the stored procedure was run using SQL Server tools and that trace file is in the shared folder named `SQLTraceFolder` on the database server named `Server2`.

Note that in this scenario, the value for `DataSourceID` parameter can be any valid CIMS Data Source for a SQL Server database on any computer. The data source does not need to point to the database that contains the stored procedure.

```
<Process id="MSSQL2000"
description="Process for SQL Server 2000 Collection"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 MSSQL2000"
type="ConvertToCSR"
programName="MSSQLServer\2000\MSSQL2000.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server2"/>
        <Parameter TraceFolder="\\Server2\SQLTraceFolder"/>
        <Parameter DataSourceID="CIMSServer"/>
        <Parameter RunSP="false"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan MSSQL2000"
type="Process"
programName="Scan"
programType="net"
active="true">
    </Step>
    <Step id="Process"
description="Standard Processing for MSSQL2000"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
    <Step id="DatabaseLoad"
description="Database Load for MSSQL2000"
type="Process"
programName="DBLoad"
programType="com"
active="true">
    </Step>
  </Steps>
```

```
<Step id="Cleanup"
      description="Cleanup MSSQL2000"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
  </Parameters>
</Step>
</Steps>
</Process>
```

For a description of each of the parameters for the `MSQL2000.wsf` script, see [Table 4-3](#) on page 4-16. For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

### Job File Example 3: Run Stored Procedures Using SQL Server Tools and Transfer Files to CIMS Data Collectors Server

As with the preceding job file example on [page 4-12](#), this example does not run the CIMSSp\_SQLServer2000Trace stored procedure. This example assumes that the stored procedure was run using SQL Server tools.

However, unlike the preceding job file example, the trace file is not pulled from the trace file folder on the database server. A copy of the trace file is transferred from the trace file folder on the database server to the trace file folder on the central CIMS Data Collectors server. Note the following:

- The log file names for the FileTransfer from parameter includes the wildcard character \* after %LogDate\_End% because the trace file names are in the format `yyyymmdd-hhmmss.trc`.
- The value for DataSourceID parameter can be any valid CIMS Data Source for a SQL Server database on any computer. The data source does not need to point to the database that contains the stored procedure.

```
<Process id="MSSQL2000"
description="Process for SQL Server 2000 Collection"
active="true">
  <Steps>
    <Step id="FileTransfer"
description="Transfer SQL Trace Files"
type="Process"
programName="FileTransfer"
programType="net"
active="true">
      <Parameters>
        <Parameter type="Windows"/>
        <Parameter from="\\Server2\SQLTraceFolder\%LogDate_End%*.trc"
to="\\Server1\SQLTraceFolder"
action="Copy"
overwrite="true"/>
      </Parameters>
    </Step>
    <Step id="Server1 Collection"
description="Server1 MSSQL2000"
type="ConvertToCSR"
programName="MSSQLServer\2000\MSSQL2000.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server2"/>
        <Parameter TraceFolder="\\Server1\SQLTraceFolder"/>
        <Parameter DataSourceID="CIMSServer"/>
        <Parameter RunSP="false"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

```
<Step id="Scan"
      description="Scan MSSQL2000"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
</Step>
<Step id="Process"
      description="Standard Processing for MSSQL2000"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
</Step>
<Step id="DatabaseLoad"
      description="Database Load for MSSQL2000"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
</Step>
<Step id="Cleanup"
      description="Cleanup MSSQL2000"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
  </Parameters>
</Step>
</Steps>
</Process>
```

For a description of each of the parameters for the `MSQQL2000.wsf` script, see [Table 4-3](#) on page 4-16. For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

## MSSQL2000.wsf Script Parameter Description

Parameter	Description/Values
LogDate	<p>The log date specifies the date for the trace file that you want to collect. For more information about using a log date, including valid log date values, see <i>Specifying Log Dates for Collection</i> on page 2-3.</p> <p><b>Note:</b> The first time that you run the job file for the SQL Server 2000 collector, the job will fail if there are no existing files in the trace folder. However, a trace file with the run date in the file name is created as a result of the job run. If you want to ensure that the job file runs correctly, run the job file again with the RNDATE keyword.</p> <p>For all subsequent runs of the job file, use the appropriate log date value as described in <i>Specifying Log Dates for Collection</i>.</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the trace file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <i>OutputFolder</i> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <i>Process id</i> attribute in the job file. For example, if the <i>Process id</i>="MSSQL2000", the output folder is MSSQL2000.</p>
TraceFolder	The location of the .trc file to be processed. A UNC is recommended.

Table 4-3 • MSSQL2000.wsf Parameters



Parameter	Description/Values
DataSourceID	<p>If you are running the CIMSp_SQLServer2000 stored procedure from a job file, this is the CIMS Data Source ID for the database that contains the stored procedure.</p> <p>If you <i>are not</i> running the stored procedure from a job file, this is any valid CIMS Data Source ID for a SQL Server database on any computer.</p>
RunSp (optional)	<p>If this parameter is set to <code>true</code>, is not included, or is left blank, the CIMSp_SQLServer2000 stored procedure is run.</p> <p>If this parameter is set to <code>false</code>, the stored procedure is not run.</p>

**Table 4-3 • MSSQL2000.wsf Parameters (Continued)**

## Running the SQL Server 2000 Collector

Use CIMS Job Runner to run the SQL Server 2000 collector as described in *Running CIMS Job Runner* on page 2-93.

## Oracle Data Collector

---

**Note** • This section pertains to Oracle running on the Windows operating system. If you are using CIMS Data Collector for UNIX to collect data for Oracle running on the UNIX operating system, see *Chapter 12, UNIX Data Collectors*.

---

The CIMS Data Collector for Oracle (called CIMSWIND) collects data from the event log and from a data file created by the CIMS Oracle Accounting Service. The event log and data file provide useful metrics such as:

- System, user, and database name.
- Amount of CPU time used by an Oracle session.
- Memory used in the User Global Area and Program Global Area.
- Number of commits performed by the user.
- Number of reads from and writes to the database files.

The following sections provide steps for setting up and running the Oracle collector and enabling Oracle logging using the CIMS Oracle Accounting Service.

### Setting Up the CIMSWIND Collector

---

**Note** • This section provides steps for setting up the CIMSWIND collector for Oracle and DB2 data collection.

---

If you are running CIMSWIND in a client/server environment, you need to install CIMSWIND on the client(s) and the server. The following post-installation instructions are applicable to both client and server unless noted otherwise. These instructions assume that CIMSWIND is installed in the default location `C:\Program Files\CIMSLab\Collectors`.

- 1 Verify that the following system environment variables have been established:

ARSAP\_DATA=C:\PROGRA~1\CIMSLab\Collectors\CIMSWIND\DATA

ARSAP\_HELP=C:\PROGRA~1\CIMSLab\Collectors\CIMSWIND\HELP

ARSAP\_HOME=C:\PROGRA~1\CIMSLab\Collectors\CIMSWIND

ARSAP\_LOG=C:\PROGRA~1\CIMSLab\Collectors\CIMSWIND\LOG

For Windows NT Server, click **Control Panel** ▶ **System** ▶ **Environment tab**.

For Windows 2000 Server, click **Control Panel** ▶ **System** ▶ **Advanced tab** ▶ **Environment Variables**.

- 2 In the `CIMSWIND\Data` folder, rename the file `Sample_NT_config_par.bat` to `NT_config_par.bat`. You need to modify this configuration file and renaming the file prevents it from being overwritten when you upgrade to a new version of CIMS Data Collectors.

- 3 Load the licensing information from the license PAK provided by CIMS Lab as follows:
  - a At the command prompt, run `CIMSWIND\Etc\NT_add_license.bat`.
  - b When prompted, enter the values contained in the license PAK exactly as provided by CIMS Lab. The values are case-sensitive.

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

- 4 Verify that the security options for the following audit policies have been set to Success/Failure:
  - For Windows NT Server, click **Start ▶ Programs ▶ Administrative Tools ▶ User Manager Policies ▶ Audit**.
    - Logon and Logoff
    - Restart, Shutdown, and System
    - Process Tracking
  - For Windows 2000 Server, click **Start ▶ Programs ▶ Administrative Tools ▶ Local Security Policy ▶ Local Policies ▶ Audit Policy**.
    - Audit logon events
    - Audit process tracking
    - Audit system events
- 5 In the Windows Event Viewer, verify that the maximum log size for all event logs is set to a size sufficient to hold event records for more than one day. This size may vary depending on the usage on any particular platform. The default setting of 512 KB is usually sufficient.
- 6 In Windows Task Scheduler, schedule the following scripts:
  - `CIMSWIND\Etc\NT_nightly.bat`. This nightly collection script should be scheduled to run nightly around 1 a.m. This script calls `NT_arsap_nightly.bat`, which produces the CIMS WIND Accounting File. The CIMS WIND Accounting File contains the combined data collected from the event log and database data file.

Note that in a client/server environment, this script is not required on the CIMS WIND server unless you are collecting data from the server.
  - `CIMSWIND\Etc\NT_process.bat`. This nightly consolidation script should be scheduled only on the CIMS WIND server and not on clients. This script calls `NT_process_nightly.bat`, which consolidates the collected CIMS WIND Accounting Files and produces CSR files. This script should be scheduled to run nightly around 5 a.m.

**7** On the CIMSWIND server, do the following:

- In CIMSWIND\Accounting, create a folder for each client computer. If this is a stand-alone implementation, create a folder for this server. This folder is used to store the CIMSWIND Accounting Files (file name acc\_yyyymmdd.dat).

The folder must have the same name as the client or server name. For example, if you are creating a folder for a client computer named ClientB, the folder name must be ClientB.

- Open CIMSWIND\Data\A\_node.par and add name of each client. If this is stand-alone implementation, add the name of this server. Enter the names on separate lines.

**8** Set the following environment variable values in NT\_config\_par.bat:

- set ARSAP\_SERVER=<server name>

If you are setting this value on a client, this is the name of the CIMSWIND server. If you are setting this value on a stand-alone server, this is the name of the server.

- set DEST=<destination path>

This variable sets the path for the destination folder for the CIMSWIND Accounting Files on the CIMSWIND server. These are the folders that you created in [Step 7](#).

If you are using CIMSWIND in a stand-alone environment, the CIMSWIND Accounting Files are stored in a folder with the same name as the server. For example, if the server name is ServerA, the environment variable would be DEST=C:\PROGRA~1\CIMSLAB\Collectors\CIMSWIND\Accounting\ServerA.

If you are using CIMSWIND in a client/server environment, the CIMSWIND Accounting Files are stored in a folder with the same name as the client. On the client, a UNC is recommended for the folder path. For example, if the client name is ClientB, the environment variable might be DEST=\\SERVERA\CIMSWIND\_ACCOUNTING\CLIENTB where the share CIMSWIND\_ACCOUNTING was created for the C:\PROGRA~1\CIMSLAB\Collectors\CIMSWIND\Accounting on ServerA.

The ARSAP\_SERVER and DEST settings are commented by default. Make sure that you remove the comment.

## Creating a Process Definition Folder for Oracle Data Collection

You need to create a process definition folder and script for the Oracle collector in the Processes folder. This folder will be used to store the CSR files created by the collector (see *About the Processes Folder* on page 2-12).

## Enabling Oracle Logging

To enable logging for Oracle, you need to use the CIMS Oracle Accounting Service. The following are instructions for starting this service for one Oracle instance. If you have multiple Oracle instances on your computer, you need to repeat these steps for each instance.

The following instructions assume that CIMSWIND is installed in the default installation location C:\Program Files\CIMSLab\Collectors and that accounting for an Oracle instance named ORCL is being enabled.

- 1 Copy the CIMS Oracle Accounting Service executable CIMSWIND\BIN\NT\_dbao.exe to create a new executable named NT\_dbao\_ORCL.exe (includes the name of the Oracle instance to be tracked).
- 2 Install the CIMS Oracle Accounting Service in Windows Services. At the command prompt, go to C:\Program Files\CIMSLab\Collectors\CIMSWIND\Bin and execute the command:

```
NT_dbao_ORCL -install
```

Note that the service can be removed from Services with the command:

```
NT_dbao_ORCL -remove
```

- 3 Create an Oracle user account to be used by the CIMS Oracle Accounting Service for connecting to the Oracle instance. In the following example, the user name is cims and the password is acct123:

```
SQL> CREATE USER cims IDENTIFIED BY acct123;
```

- 4 The Oracle user cims must be able to select the following ORACLE instance V\$ system tables:

```
V$DATABASE
```

```
V$PROCESS
```

```
V$SESSION
```

```
V$SESSTAT
```

```
V$STATNAME
```

An SQL script, CIMSWIND\Etc\Oracle\arsap\_view.sql, is included in the installation. This script creates an Oracle role called ARSAP\_VIEW with the necessary privileges. The script grants the role to the Oracle user cims. The Oracle DBA can run this script after the Oracle user has been created.

- 5 Create a CIMS DB Instance Record for this Oracle instance. To create this record, run the CIMSWIND setup utility, CIMSWIND\Bin\NT\_setup.exe.

At the SETUP> prompt, enter the command:

```
SETUP>add/dbinst/dbtype=ORACLE/user=cims/password=acct123/frequency=60 ORCL
```

Where 60 indicates a sample frequency of every 60 seconds.

- 6 Start the CIMS ORACLE Accounting Service either from Services or from the following command at the command prompt:

```
NET START "CIMS/NT Oracle DB Collector-ORCL"
```

- 7 In the CIMSWIND\Data\NT\_config\_par.bat script, set the following environment variables:

- set A\_ORACLE\_ACCT=Y

Setting this variable to Y instructs the NT\_arsap\_nightly.bat script to include the CIMSWIND Oracle Accounting File in the files collected nightly.

- set GEN\_ORACLE=Y

Setting this variable to Y instructs the NT\_process\_nightly.bat script to generate Oracle CSR files for input into CIMS Server.

- set CS\_GCS\_DEST=<destination folder for CSR files>

This is the destination folder for the generated CSR files. You need to change this location to the job process folder that you created in *Creating a Process Definition Folder for Oracle Data Collection* on page 4-21.

---

**Note** • NT\_config\_par.bat is shipped as Sample\_NT\_config\_par.bat. You should have renamed the script in [Step 2](#) on page 4-18.

---

- 8 Schedule the script CIMSWIND\Etc\CIMS\_start\_db\_svc.bat to be run when the computer is started. This script automatically starts the CIMS Oracle Accounting Service.

This script requires modification as indicated in the comments at the beginning of the script. Rename this script so that the modification is not overwritten when you upgrade to a new version of CIMS Data Collectors.

## Resources Collected

By default, the following resources in the event log and Oracle data file are defined as the chargeback resources in CIMS Server.

Resource	Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Event Log Resources</b>		
Logins to the system	MS Windows Logins	LLT101
Connect time on the system in hours	MS Windows Connect Time (hours)	LLT102
Number of images executed	MS Windows Image Count	LLT103
Time spent executing	MS Windows Image Time (hours)	LLT104
<b>Oracle Resources</b>		
Number of Oracle sessions	MS Windows Oracle Logins	LLW101
CPU utilized in Oracle sessions	MS Windows Oracle Session CPU (minutes)	LLW102
Amount of time a user is connected to Oracle	MS Windows Oracle Connect (hours)	LLW103
Memory used in the User Global Area	MS Windows Oracle UGA Memory	LLW104
Memory used in the Program Global Area	MS Windows Oracle PGA Memory	LLW105
Oracle Recursive CPU – CPU used updating internal tables	MS Windows Oracle Rec CPU (minutes)	LLW106
Commits performed by the user	MS Windows Oracle User Commits	LLW107
Reads from database files	MS Windows Oracle Physical Reads	LLW108
Writes to database files	MS Windows Oracle Physical Writes	LLW109
Write requests to database files	MS Windows Oracle Write Requests	LLW110
Memory utilized to perform an external sort	MS Windows Oracle Disk Sorts	LLW111

**Table 4-4 • Default Oracle Resources**

Resource	Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
Messages sent to perform database updates	MS Windows Oracle Messages Sent	LLW112
Messages received to update database	MS Windows Oracle Messages Received	LLW113

**Table 4-4 • Default Oracle Resources (Continued)**

## Setting Up the Oracle Collector Job File

On the CIMSWIND server, set up an XML job file as described in [Creating Job Files](#) on page 2-25. The job file must contain a `Process` element for the process definition folder that you created [Creating a Process Definition Folder for Oracle Data Collection](#) on page 4-21. Note that the `ConvertToCSR` step is not required because the process definition folder contains CSR files created by the `NT_process.bat` script.

```

<Process>    id="WinOracle"
             description="Windows Oracle Collection"
             active="true">
  <Steps>
    <Step    id="Scan WinOracle"
             description="Scan WinOracle"
             type="Process"
             programName="Scan"
             programType="net"
             active="true">
    </Step>
    <Step    id="Process"
             description="Standard Processing for WinOracle"
             type="Process"
             programName="SingleProcessStep"
             programType="com"
             active="true">
    </Step>
    <Step    id="DatabaseLoad"
             description="Database Load for WinOracle"
             type="Process"
             programName="DBLoad"
             programType="com"
             active="true">
    </Step>
    <Step    id="Cleanup"
             description="Cleanup WinOracle"
             type="Process"
             programName="Cleanup"
             programType="net"
             active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```



## Running the Oracle Collector

To run the Oracle collector, you need to run the following scripts and program:

- C:\CIMS\WIND\Etc\NT\_nightly.bat. This script should be scheduled to run nightly around 1 a.m. Note that in a client/server environment, this script is not required on the CIMS\WIND server unless you are collecting data from the server.
- C:\CIMS\WIND\Etc\NT\_process.bat. This script should be scheduled to run nightly around 5 a.m. The script should be scheduled only on the CIMS\WIND server and not on clients.
- CIMS Job Runner. This program should be scheduled to run nightly after NT\_process.bat has run. For instructions for running CIMS Job Runner, see *Running CIMS Job Runner* on page 2-93.

Make sure that the CIMS\WIND collector is set up correctly as described on *Setting Up the CIMS\WIND Collector* on page 4-18 and that the job file is set up as described in *Creating Job Files* on page 2-25.

## DB2 Data Collector

---

**Note** • This section pertains to DB2 running on the Windows operating system. If you are using CIMS Data Collector for UNIX to collect data for DB2 running on the UNIX operating system, see [Chapter 12, UNIX Data Collectors](#).

---

The CIMS Data Collector for DB2 (called CIMSWIND) collects data from the event log and from a data file created by the CIMS DB2 Accounting Service. The event log and data file provide useful metrics such as:

- System, user, and database name.
- System and user CPU utilization.
- Number of read and write operations that do not use a buffer pool.
- Buffered pool data writes and logical and physical reads.
- Buffered pool index writes and logical and physical reads.
- Number of row delete, insert, and update operations.

The following sections provides instructions for creating a DB2 process definition folder, running the DB2 collector, and enabling DB2 logging.

You also need to complete the instructions for setting up the CIMSWIND collector that are provided in the [Oracle Data Collector](#) section. These instructions are applicable to both Oracle and DB2.

### Setting up the CIMSWIND Collector

See [Setting Up the CIMSWIND Collector](#) on page 4-18.

### Creating a Process Definition Folder for DB2 Data Collection

You need to create a process definition folder and script for the DB2 collector in the Processes folder. This folder will be used to store the CSR files created by the collector (see [About the Processes Folder](#) on page 2-12).

## Enabling DB2 Logging

To enable logging for DB2, you need to use the CIMS DB2 Accounting Service. The following are instructions for starting this service for one DB2 instance. If you have multiple DB2 instances on your computer, you need to repeat these steps for each instance.

The following instructions assume that CIMSWIND is installed in the default installation location `C:\Program Files\CIMSLab\Collectors\CIMSWIND` and that accounting for a DB2 instance named `DB2MPP` is being enabled.

- 1 Copy the CIMS DB2 Accounting Service executable `CIMSWIND\BIN\NT_dbadb2.exe` to create a new executable named `NT_dbadb2_DB2MPP.exe` (includes the name of the DB2 instance to be tracked).
- 2 Install the CIMS DB2 Accounting Service in Windows Services. At the command prompt, go to `C:\Program Files\CIMSLab\Collectors\CIMSWIND\Bin` and execute the command:

```
NT_dbadb2_DB2MPP -install
```

Note that the service can be removed from Services with the command:

```
NT_dbadb2_DB2MPP -remove
```

- 3 Set the following DB2 Monitor switches to on for the instance:
  - BUFFERPOOL
  - LOCK
  - SORT
  - UOW
- 4 Create a CIMS DB Instance Record for this DB2 instance. To create this record, run the CIMSWIND setup utility, `CIMSWIND\Bin\NT_setup.exe`.

At the `SETUP>` prompt, enter the command as shown in the following example where a user named `cims` has sufficient privileges to access DB2 monitoring information. The password for this user is `acct123` and the sample frequency is every 60 seconds.

```
SETUP>add/dbinst/dbtype=DB2/user=cims/password=acct123/frequency=60 DB2MPP
```

- 5 Start the CIMS DB2 Accounting Service either from Services or from the following command at the command prompt:

```
NET START "CIMS/NT DB2 Collector-DBMPP"
```

6 In the CIMSWIND\Data\NT\_config\_par.bat script, set the following environment variables:

- set A\_DB2\_ACCT=Y

Setting this variable to Y instructs the NT\_arsap\_nightly.bat script to include the CIMSWIND DB2 Accounting File in the files collected nightly.

- set GEN\_DB2=Y

Setting this variable to Y instructs the NT\_process\_nightly.bat script to generate DB2 CSR files for input into CIMS Server.

- set CS\_GCS\_DEST=<destination folder for CSR files>

This is the destination folder for the generated CSR files. You need to change this location to the job process folder that you created in [Creating a Process Definition Folder for DB2 Data Collection](#) on page 4-26.

---

**Note** • NT\_config\_par.bat is shipped as Sample\_NT\_config\_par.bat. You should have renamed the script in [Step 2 on page 4-18](#).

---

7 Schedule the script CIMSWIND\Etc\CIMS\_start\_db\_svc.bat to be run when the computer is started. This script automatically starts the CIMS DB2 Accounting Service.

This script requires modification as indicated in the comments at the beginning of the script. Rename this script so that the modification is not overwritten when you upgrade to a new version of CIMS Data Collectors.

## Resources Collected

By default, the following resources in the event log and DB2 data file are defined as the chargeback resources in CIMS Server.

Resource	Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Event Log Resources</b>		
Logins to the system	MS Windows Logins	LLT101
Connect time on the system in hours	MS Windows Connect Time (hours)	LLT102
Number of images executed	MS Windows Image Count	LLT103
Time spent executing	MS Windows Image Time (hours)	LLT104
<b>DB2 Resources</b>		
SQL commit statements that have been attempted	MS Windows DB/2 Commit SQL STMTS	LLX101
Number of deadlocks that have occurred	MS Windows DB/2 Deadlocks	LLX102
The number of read operations that do not use the buffer pool	MS Windows DB/2 Direct Reads	LLX103
The number of write operations that do not use the buffer pool	MS Windows DB/2 Direct Writes	LLX104
Rollbacks initiated by the database manager due to a deadlock	MS Windows DB/2 Int Deadlock Rollbacks	LLX105
Elapsed time waiting for a lock	MS Windows DB/2 Lock Wait Time	LLX106
The number of times a user connects to the database	MS Windows DB/2 Logins	LLX107
Buffered pool data logical reads	MS Windows DB/2 PD Lreads	LLX108
Buffered pool data physical reads	MS Windows DB/2 PD Preads	LLX109

**Table 4-5 • Default DB2 Resources**

Resource	Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
Buffered pool data writes	MS Windows DB/2 PD Writes	LLX110
Buffered pool index logical reads	MS Windows DB/2 PI Lreads	LLX111
Buffered pool index physical reads	MS Windows DB/2 PI Preads	LLX112
Buffered pool index writes	MS Windows DB/2 PI Writes	LLX113
SQL rollback statements attempted	MS Windows DB/2 Rollback SQL Statements	LLX114
The number of row deletion operations	MS Windows DB/2 Rows Deleted	LLX115
The number of row inserted operations	MS Windows DB/2 Rows Inserted	LLX116
The number of row select/returned to the application	MS Windows DB/2 Rows Selected	LLX117
The number of row updated operations	MS Windows DB/2 Rows Updated	LLX118
System CPU used by the database manager process	MS Windows DB/2 SCPU (minutes)	LLX119
Number of sorts that ran out of sort heap	MS Windows DB/2 Sort Overflows	LLX120
Number of sorts executed	MS Windows DB/2 Total Sorts	LLX121
LLX122	MS Windows DB/2 UCPU (minutes)	User CPU used by the database manager process
LLX123	MS Windows DB/2 UOW Log Space Used	The amount of log space (in bytes) used in the current unit

**Table 4-5 • Default DB2 Resources (Continued)**

## Setting Up the DB2 Collector Job File

On the CIMSWIND server, set up an XML job file as described in [Creating Job Files](#) on page 2-25. The job file must contain a `Process` element for the process definition folder that you created [Creating a Process Definition Folder for DB2 Data Collection](#) on page 4-26.

Note that the `ConvertToCSR` step is not required because process definition folder contains CSR files created by the `NT_process.bat` script.

```
<Process>    id="WinDB2"
             description="Windows DB2 Collection"
             active="true">
  <Steps>
    <Step    id="Scan WinDB2"
            description="Scan WinDB2"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
      </Step>
    <Step    id="Process"
            description="Standard Processing for WinDB2"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
      </Step>
    <Step    id="DatabaseLoad"
            description="Database Load for WinDB2"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
      </Step>
    <Step    id="Cleanup"
            description="Cleanup WinDB2"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

## Running the DB2 Collector

To run the DB2 collector, you need to run the following scripts and programs:

- C:\CIMSWIND\Etc\NT\_nightly.bat. This script should be scheduled to run nightly around 1 a.m. Note that in a client/server environment, this script is not required on the CIMSWIND server unless you are collecting data from the server.
- C:\CIMSWIND\Etc\NT\_process.bat. This script should be scheduled to run nightly around 5 a.m. The script should be scheduled only on the CIMSWIND server and not on clients.
- CIMS Job Runner. This program should be scheduled to run nightly after NT\_process.bat has run. For instructions for running CIMS Job Runner, see [Running CIMS Job Runner](#) on page 2-93.

Make sure that the CIMSWIND collector is set up correctly as described on [Setting Up the CIMSWIND Collector](#) on page 4-18 and that the job file is set up as described in [Creating Job Files](#) on page 2-25.



## Sybase Data Collector

CIMS Lab provides a CIMS Data Collector for Sybase. For instructions on how to configure this collector, contact CIMS Lab.

## Database Size Data Collector (DBSpace)

The CIMS Data Collector for database size, DBSpace, collects data regarding the size of all Microsoft SQL Server or Sybase databases on a server. The DBSpace collector uses the stored procedure `sp_helpdb`. To run this collector, you need authority to run `sp_helpdb`.

The following sections provide instructions for setting up and running the DBSpace collector.

### Identifiers and Resources Collected by the DBSpace Collector

By default, the following data collected by the DBSpace collector is defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `DBSpace.wsf` conversion script).

The rate code assigned to the SQL Server database size resource (`MSDBSIZE`) is pre-loaded in the `CIMSRate` table. The rate code assigned to the Sybase database size (`SYDBSIZE`) is *not* pre-loaded and must be added to the `CIMSRate` table as described in the *CIMS Server Administrator's Guide*.

#### Identifiers

- Feed (this is passed from DBSpace collector job file)
- Database
- Owner
- DBID (Database ID)

#### Resource Rate Codes

- `MSDBSIZE` (MS Windows SQL Server Used [MB Days])
- `SYDBSIZE` (Sybase database size in megabytes)

## Setting Up the DBSpace Collector

### Creating a CIMS Data Source ID

You need to create a CIMS Data Source for any database in a SQL Server instance that you want to collect data from. The collector will collect data from all databases in the instance.

For the steps required to create a CIMS Data Source ID, see [Appendix B](#).

## Set Up the Job File

On the central CIMS Data Collectors server, set up an XML job file for the DBSpace collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file:

```

<Process id="DBSpace"
  description="Process for DBSpace Collection"
  active="true">
  <Defaults>
    <Default LogDate="RNDATE"/>
  </Defaults>
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 DBSpace"
      type="ConvertToCSR"
      programName="DBSpace\DBSpace.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter DBType="MS"/>
        <Parameter DataSourceID="DBSpace"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan DBSpace"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for DBSpace"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for DBSpace"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup DBSpace"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the DBSpace collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 4-6](#). These parameters are used by the conversion script, `DBSpace.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The DBSpace collector collects data that is current as of the date and time that the collector is run by CIMS Job Runner. However, the start and end date that appears in the CSR file records and the date that appears in the initial CSR file name will reflect the value entered for the <code>LogDate</code> parameter. For example, if you use the <code>LogDate</code> parameter <code>PREDAY</code>, the previous day's date is used.</p> <p>To include the actual date that the data was collected, you need to include the parameter <code>LogDate="RNDATE"</code> at the job or process level in the job file (see the example on <a href="#">page 4-34</a>).</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the databases that you want to collect size data from.</p> <p>A subfolder with the same name as the computer is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="DBSpace"</code>, the output folder is <code>DBSpace</code>.</p>

**Table 4-6 • DBSpace.wsf Parameters**

## ■ Database Data Collectors

### Database Size Data Collector (DBSpace)

---

Parameter	Description/Values
DBType	The database type. Valid values are: <ul style="list-style-type: none"><li>■ MS (SQL Server)</li><li>■ SY (Sybase)</li></ul>
DataSourceID	The CIMS Data Source ID for any database in a SQL Server instance that you want to collect data from. The collector will then collect data from all databases in the instance.

**Table 4-6 • DBSpace.wsf Parameters (Continued)**

## Running the DBSpace Collector

Use CIMS Job Runner to run the DBSpace collector as described in *Running CIMS Job Runner* on page 2-93.

## SQL Server Reporting Services Data Collector

The CIMS Data Collector for SQL Server Reporting Services collects data that is contained in the Report Server Execution Log. This log contains information about the reports that execute on the server or on multiple servers in a single Web farm and log provides the following useful metrics:

- Name of the report server instance that handled the request.
- Report and user identifier.
- Report format.
- Parameter values used for a report execution.
- Source of the report execution.
- Time spent retrieving the data, processing the report, and rendering the report.
- Size of rendered reports in bytes.
- Number of rows returned from queries.

This collector uses the same components as the Universal collector as described in [Chapter 14, CIMS Universal Data Collector](#).

## Identifiers and Resources Collected by the Reporting Services Collector

By default, the following values in the Report Server Execution Log are defined as chargeback identifiers and resource rate codes.

### Identifiers

- InstanceName
- ReportID
- UserName
- Format
- Parameters
- Source

### Resources

- RSTIMRET (time spent retrieving data)
- RSTIMPRO (time spent processing report)
- RSTIMREN (time spent rendering report)
- RSBYTES (size of report in bytes)
- RSROWS (number of rows returned from queries)

These identifiers and resources are defined in the conversion definition file `ExecutionLogDef.txt`. This file contains the conversion information required by CIMS Conversion Engine to create a CSR file from the Report Server Execution Log. If you installed CIMS Server in the default location, this file is in `C:\Program Files\CIMSLab\Collectors\MSReportingServices`.

**Note** • You will need to modify the `ExecutionLogDef.txt` file to point to the ODBC Data Source for the database that contains the Report Server Execution Log. To modify the `ExecutionLogDef.txt` file to point to the correct data source, use the CIMS Conversion Builder GUI (see *Creating a Conversion Definition Using CIMS Conversion Builder* on page 14-3).

The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

## Setting Up the Reporting Services Collector

On the central CIMS Data Collectors server, set up an XML job file for the Reporting Services collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file. Because this collector uses CIMS Conversion Engine, the conversion script for the Universal collector (`Universal.wsf`) is called.

```
<Process id="Reporting Services"
description="Process for Reporting Services"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 Reporting Services"
type="ConvertToCSR"
programName="Universal\Universal.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter ConvEngDefName="C:\Program Files\CIMSLab\Collectors\
MSReportingServices\ExecutionLogDef.txt"/>
        <Parameter OutputFolder="%ProcessFolder%"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan Reporting Services"
type="Process"
programName="Scan"
programType="net"
active="true">
  </Step>
    <Step id="Process"
description="Standard Processing for Reporting Services"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
  </Step>
    <Step id="DatabaseLoad"
```

```

        description="Database Load for Reporting Services"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup Reporting Services"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Universal collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 4-7](#). These parameters are used by the conversion script, `Universal.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The log date specifies the data that appears in the initial CSR file name. This is the CSR file that is processed by the Scan program. The start and end dates that appear in the CSR file records are defined by the definition file <code>ExecutionLogDef.txt</code>.</p> <p>For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the Report Server Execution Log.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p>

**Table 4-7 • Universal.wsf Parameters**

Parameter	Description/Values
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="MSReportingServices"</code>, the output folder is <code>MSReportingServices</code>.</p>
ConvEngDefName	<p>The location of the conversion definition file <code>ExecutionLogDef.txt</code>.</p>

---

**Table 4-7 • Universal.wsf Parameters (Continued)**



# E-mail Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for e-mail applications. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Microsoft Exchange Server 5.5 Collector</b> .....	<b>5-2</b>
Enabling Exchange Server 5.5 Logging .....	5-2
Exchange Server 5.5 Log File Name and Format .....	5-3
Identifiers and Resources Collected from the Exchange Server 5.5 Log File .....	5-5
<b>Microsoft Exchange Server 2000/2003 Collectors</b> .....	<b>5-6</b>
Enabling Exchange Server 2000/2003 Logging .....	5-6
Exchange Server 2000/2003 Log File Name and Format .....	5-7
Identifiers and Resources Collected from the Exchange Server 2000/2003 Log File .....	5-9
<b>Setting Up and Running the Exchange Server Collectors</b> .....	<b>5-12</b>
Setting Up the Exchange Server Collectors .....	5-12
Running the Exchange Server Collectors .....	5-14
<b>Microsoft Exchange Server Mailbox 5.5, 2000, and 2003 Data Collector</b> .....	<b>5-15</b>
Requirements .....	5-15
Troubleshooting .....	5-16
Identifiers and Resources Collected from the Exchange Server Mailbox Store .....	5-17
Setting Up the Exchange Server Mailbox Collector .....	5-20
Running the Exchange Server Mailbox Collector .....	5-23
<b>Microsoft Outlook Web Access Data Collection</b> .....	<b>5-24</b>
<b>Lotus Notes Data Collector</b> .....	<b>5-24</b>

## Microsoft Exchange Server 5.5 Collector

The CIMS Data Collector for Microsoft Exchange Server 5.5 collects and processes data that is contained in a log file produced by Exchange Server. This log file provides useful metrics such as the number of e-mail messages and bytes sent and received by user.

The following sections provide the following information:

- Instructions for enabling logging for Exchange Server 5.5.
- A description of the fields in the Exchange Server log file.
- A description of the identifiers and resources that are collected from the log file.

For the instructions on how to set up and run any Exchange Server collector (5.5, 2000, or 2003), see *Setting Up and Running the Exchange Server Collectors* on page 5-12.

### Enabling Exchange Server 5.5 Logging

The following provides an example of enabling message tracking logging for Exchange Server 5.5 components on Windows NT Server 4.0. Refer to the Microsoft documentation for instructions on how to enable logging on other platforms.

#### Enabling Message Tracking on MTAs

To enable message tracking on all Message Transfer Agents (MTAs) on a site:

- 1 In the Microsoft Exchange Administrator window, click **Configuration** or **Information Site Configuration**.
- 2 Double-click **MTA Site Configuration**.

The MTA Site Configuration Properties dialog box appears.

- 3 On the **General** tab, select the **Enable message tracking** check box.
- 4 Restart the MTAs or restart the computer.

#### Enabling Message Tracking on a Microsoft Mail Connector

You must enable message tracking separately on each mail connector on a site. To enable message tracking:

- 1 In the Microsoft Exchange Administrator window, click **Connections**.
- 2 Double-click a mail connector.

The connector properties dialog box appears.

- 3 On the **Interchange** tab, select the **Enable message tracking** check box.
- 4 Restart the mail connector or restart the computer.

## Enabling Message Tracking on the Internet Mail Service

You must enable message tracking separately on each Internet Mail Service on a site. To enable message tracking:

- 1 In the Microsoft Exchange Administrator window, navigate to and click **Connections**.
- 2 Double-click an Internet Mail Service.

The Internet Mail Service Properties dialog box appears.

- 3 On the **Internet Mail** tab, select the **Enable message tracking** check box.
- 4 Restart the Internet Mail Service or restart the computer.

For more information about Exchange Server 5.5 logging, refer to the Microsoft documentation.

## Exchange Server 5.5 Log File Name and Format

The Exchange Server 5.5 message tracking log is stored in `exchsrvr\tracking.log`. Each day, a new log is created that records one day's activities on the server. Each daily log is named by the date on which it was created in `yyyymmdd.log` format.

The following table describes the record fields in the Exchange Server 5.5 log file.

Field Name	Description/Values
Message ID or MTS-ID	<p>Message ID is a unique identifier assigned to the message by Exchange Server. It stays with the message from its origination to delivery or transfer from the network.</p> <p>Messages from foreign systems include a message transfer system-ID (MTS-ID) that uniquely identifies the component that transported the message.</p>
Event #	A number that represents the event type.
Date/Time	Date and time of the event.
Gateway Name	Name of the gateway or connector that generated the event. If no gateway was involved, the field is blank.
Partner Name	Name of the messaging service associated with the event. In Exchange Server, the partner name is the MTA or Information Store.
Remote ID	Message ID used by the gateway.
Originator	Distinguished name of the originating mailbox, if known.

**Table 5-1 • Exchange Server 5.5 Log File Format**

Field Name	Description/Values
Priority	Priority set by the sender. 0 = Normal 1= High -1 = Low
Length	Message length in bytes.
Seconds	Transport time in seconds. Not used by Exchange Server. The value in this field is 0 or blank.
Cost	Cost per second for message transfer. Not used by Microsoft Exchange Server. The value in this field is always 1.
Subject-ID or Report MTS-ID	This field is blank (empty) for normal messages. For reports its value is the Report MTS-ID.
Recipients	Number of recipients.
Recipient Name	Distinguished name of the recipient of the message or a proxy address.  This field is separated from the previous field by a line feed and is repeated for each recipient. Because this field is separated by a line, the Exchange Server collector recognizes this field as a record.
Recipient Report Status	A number representing the result of an attempt to deliver a report to the recipient.  Delivered = 0 Not delivered = 1  This is used only for reports. On other events, it is blank. This field is repeated for each recipient.

**Table 5-1 • Exchange Server 5.5 Log File Format (Continued)**

## Identifiers and Resources Collected from the Exchange Server 5.5 Log File

By default, the Exchange Server 5.5 collector gathers data from log file records that contain the following event types in the Event # field:

- 7—Message transfer out
- 9—Message delivered

Depending on the event type, the following fields in the Exchange Server log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in `MSExchange55.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

If you want to gather data for other event types, contact CIMS Lab.

Event #	Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>			
7 or 9	—	Feed (defined in the Exchange Server 5.5 collector job file)	—
7	Originator	User	—
9	Recipient Name	User	—
	<b>Note:</b> Only local recipients are collected.		
<b>Resources</b>			
7	— A value of 1 is automatically assigned.	MS Exchange Emails Sent	EXEMSNT
7	Length	MS Exchange Bytes Sent	EXBYSNT
9	— A value of 1 is automatically assigned for each local recipient record.	MS Exchange Emails Received	EXEMRCV
9	Length	MS Exchange Bytes Received	EXBYRCV

**Table 5-2 • Default Exchange Server 5.5 Identifiers and Resources**

## Microsoft Exchange Server 2000/2003 Collectors

The CIMS Data Collectors for Microsoft Exchange Server 2000 and 2003 collect and process data that is contained in a log file produced by Exchange Server. This log file provides useful metrics such as the number of e-mail messages and bytes sent and received by user.

The following sections provide the following information:

- Instructions for enabling logging for Exchange Server 2000/2003.
- A description of the fields in the Exchange Server log file.
- A description of the identifiers and resources that are collected from the log file.

The instructions for setting up and running an Exchange Server collector are the same for all Exchange Server versions (5.5, 2000, or 2003). See [Setting Up and Running the Exchange Server Collectors](#) on page 5-12.

### Enabling Exchange Server 2000/2003 Logging

The following provides an example of enabling message tracking logging for Exchange Server 2000/2003:

- 1** In the Exchange System Manager window, double-click **Server**.
- 2** Right-click a server, and then click **Properties**.
- 3** On the **General** tab, click the **Enable message tracking** check box.

For more information about Exchange Server 2000/2003 logging, refer to the Microsoft documentation.

## Exchange Server 2000/2003 Log File Name and Format

The Exchange Server 2000/2003 message tracking log is stored in `Exchsrvr\servername.log` in which `servername` is the name of your Exchange server. Each day, a new log is created that records one day's activities on the server. Each daily log is named by the date on which it was created in `yyyymmdd.log` format.

The following table describes the record fields in the Exchange Server 2000/2003 log file.

Field Name	Description/Values
Date	Date of the event.
Time	Time of the event.
client-ip	IP address of the sending client or system.
Client-hostname	Host name of the sending client system.
Partner-Name	Name of the messaging service associated with the event. In Exchange Server, the partner-name is the MTA or Information Store.
Server-hostname	Host name of the server making the log entry.
server-IP	IP address of the server making the log entry.
Recipient-Address	Name of message recipient or a proxy address.  This field is separated from the previous field by a line feed and is repeated for each recipient. Because this field is separated by a line, the Exchange Server collector recognizes this field as a record.
Event-ID	A number that represents the event type.
MSGID	Message ID is a unique identifier assigned to the message by Exchange Server. It stays with the message from its origination to delivery or transfer from the network.
Priority	Priority set by the sender.  0 = Normal 1 = High -1 = Low

**Table 5-3 • Exchange 2000/2003 Server Log File Format**

Field Name	Description/Values
Recipient-Report-Status	<p>The number of attempts required to deliver a report to the recipient, in which Delivered = 0 and Not delivered = 1.</p> <p>This field is separated from the previous field by a line feed and is repeated for each recipient. Because this field is separated by a line, the Exchange Server collector recognizes this field as a record.</p>
total-bytes	Message length in bytes.
Number-Recipients	Number of recipients.
Origination-Time	Time in seconds it took to deliver the message.
Encryption	<p>The encryption type of the message body.</p> <p>0 = No encryption            1= Message is signed            2 = Message is encrypted</p> <p>Encryption is tracked for each message, not for each recipient.</p>
service-Version	Version of the service making the log entry.
Linked-MSGID	If there is a message ID (MSGID) from another service, it is provided to link the message across services.
Message-Subject	The subject message, truncated to 106 bytes.
Sender-Address	Primary address of the originating mailbox, if known. The address can be an SMTP address, X.400 address, or a domain name, depending on the transport.

**Table 5-3 • Exchange 2000/2003 Server Log File Format (Continued)**



## Identifiers and Resources Collected from the Exchange Server 2000/2003 Log File

By default, the Exchange Server 2000/2003 collector gathers data from log file records that contain a sent or received event number in the Event ID field. The event number correlates to an event type. For a complete list of Exchange Server event numbers and types, see [page 5-10](#).

Depending on the event type, the following fields in the Exchange Server 2000/2003 log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `MSExchange<version>.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

If you want to gather data for other event types, contact CIMS Lab.

Event Type	Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>			
Sent or Received	—	Feed (defined in the Exchange 2000 Server collector job file)	—
Sent	Sender Address	User	—
Received	Recipient Address	User	—
	<b>Note:</b> Only local recipients are collected.		
<b>Resources</b>			
Sent	—	MS Exchange Emails Sent	EXEMSNT
	A value of 1 is automatically assigned.		
Sent	Total-Bytes	MS Exchange Bytes Sent	EXBYSNT
Received	—	MS Exchange Emails Received	EXEMRCV
	A value of 1 is automatically assigned for each local recipient record.		
Received	Total-Bytes	MS Exchange Bytes Received	EXBYRCV

**Table 5-4 • Default Exchange 2000/2003 Server Identifiers and Resources**

## **Exchange Server 2000/2003 Event Types**

Table 5-5 lists the Exchange Server 2000/2003 event types by their corresponding event number. For more information about the event types, refer to the following Microsoft Knowledge Base Articles:

- Exchange Server 2000  
246959 (<http://support.microsoft.com/default.aspx?scid=kb;EN-US;246959>)
- Exchange Server 2003  
821905 (<http://support.microsoft.com/default.aspx?scid=kb;en-us;821905>)

<b>Event Number</b>	<b>Event Type</b>
0	Message transfer in
1	Probe transfer in
2	Report transfer in
4	Message submission
5	Probe submission
6	Probe transfer out
7	Message transfer out
8	Report transfer out
9	Message delivered
10	Report delivered
26	Distribution list expansion (Exchange Server 2000 only)
28	Message redirected
29	Message rerouted
31	Downgrading
33	Report absorption
34	Report generation
43	Unroutable report discarded
50	Gateway deleted message
51	Gateway deleted probe

**Table 5-5 • Exchange Server 2000/2003 Event Types**

Event Number	Event Type
52	Gateway deleted report
1000	Local delivery
1001	Backbone transfer in
1002	Backbone transfer out
1003	Gateway transfer out
1004	Gateway transfer in
1005	Gateway report transfer in
1006	Gateway report transfer out
1007	Gateway report generation
1010	SMTP queued outbound
1011	SMTP transferred outbound
1012	SMTP received inbound
1013	SMTP transferred inbound
1014	SMTP message rerouted
1015	SMTP report transferred in
1016	SMTP report transferred out
1017	SMTP report generated
1018	SMTP report absorbed
1019	SMTP submitted message to advanced queuing
1020	SMTP outbound transfer
1021	SMTP bad mail
1022	SMTP advance queueing failure
1023	SMTP local delivery
1024	SMTP submit message to categorizer
1025	SMTP begin submit message
1026	SMTP advanced queuing failed message
1027	SMTP submit message to store driver

**Table 5-5 • Exchange Server 2000/2003 Event Types (Continued)**

Event Number	Event Type
1028	SMTP store driver local delivery
1029	SMTP store driver gateway delivery
1030	SMTP NDR all
1031	SMTP end outbound transfer
<b>The following event types are applicable to Exchange Server 2003 only.</b>	
1032	SMTP: Message scheduled to retry categorization
1033	SMTP: Message categorized and queued for routing
1034	SMTP: Message routed and queued for remote delivery
1035	SMTP: Message scheduled to retry routing
1036	SMTP: Message queued for local delivery
1037	SMTP: Message scheduled to retry local delivery
1038	SMTP: Message routed and queued for gateway delivery

**Table 5-5 • Exchange Server 2000/2003 Event Types (Continued)**

## Setting Up and Running the Exchange Server Collectors

This information in this section is applicable to all of the Exchange Server collectors (5.5, 2000, and 2003).

### Setting Up the Exchange Server Collectors

On the central CIMS Data Collectors server, set up an XML job file for the Exchange Server collector as described in *Creating Job Files* on page 2-25. The following is an example process for the Exchange Server 2003 collector in the job file:

```
<Process id="MSExchange"
description="Process for Exchange Server Collector"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 MSExchange"
type="ConvertToCSR"
programName="MSExchange\MSExchange2003.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

```

    <Step id="Scan"
        description="Scan MExchange"
        type="Process"
        programName="Scan"
        programType="net"
        active="true">
    </Step>
    <Step id="Process"
        description="Standard Processing for MExchange"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>
    <Step id="DatabaseLoad"
        description="Database Load for MExchange"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup MExchange"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Exchange Server collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 5-6](#). These parameters are used by the conversion script, `MExchange<version>.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.

**Table 5-6 • MExchange<version>.wsf Parameters**

Parameter	Description/Values
Feed	<p>The name of the server that contains the log file that you want to process.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id</code>="MSEExchange", the output folder is MSEExchange.</p>
LogFolder	<p>The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.</p>

**Table 5-6 • MSEExchange<version>.wsf Parameters (Continued)**

## Running the Exchange Server Collectors

Use CIMS Job Runner to run the Exchange Server collectors as described in *Running CIMS Job Runner* on page 2-93.

# Microsoft Exchange Server Mailbox 5.5, 2000, and 2003 Data Collector

The CIMS Data Collector for Microsoft Exchange Server Mailbox 5.5, 2000, and 2003 collects and processes data contained in the Exchange mailbox store. The Exchange Server Mailbox collector provides the following useful metrics:

- Mailbox count
- Mailbox size
- Number of messages in the mailbox by user

## Requirements

### LDAP V3

Microsoft Exchange Server or Active Directory must be running Lightweight Directory Access Protocol (LDAP) V3 or later.

### CDO 1.21

If you want to collect mailbox size and number of messages, the Collaboration Data Objects (CDO) 1.21 library must be installed on the central CIMS Data Collectors server. You must install CDO from another product such as Microsoft Outlook or Microsoft Exchange. For a list of products that provide CDO, refer to the Microsoft Knowledge Base Article 171440 (<http://support.microsoft.com/default.aspx?scid=kb;EN-US;171440>).

CIMS Lab has found that installing CDO from Microsoft Outlook provides the most successful results. If you use this option, you need to install Outlook on the CIMS Data Collectors server and select the Custom option during the installation set up. The Custom option enables you to select Collaboration Data Objects.

## Security Permissions

### *To collect mailbox count:*

The Windows account running the Exchange Server Mailbox collector requires Read security permission for the Exchange Server 5.5 directory or Active Directory (Exchange Server Mailbox 2000 and 2003).

### *To collect mailbox size and number of messages:*

If you want to collect mailbox size and number of messages for Exchange Server Mailbox 5.5, 2000, and 2003, you need to set additional security permissions that enable the Windows account to be able to read all mailboxes. To set these permissions, do the following:

- 1 Follow the steps in Microsoft Knowledge Base Article 262054 (<http://support.microsoft.com/default.aspx?scid=kb;EN-US;262054>).
- 2 If you use the first method in the article, add the Windows account to the Exchange Services group. The Windows account *cannot* be a member of the Administrators, Domain Admins, or Enterprise Admins group.

## Troubleshooting

The following are potential errors that you might encounter when running the Exchange Server Mailbox collector. These errors appear in the job log file.

Error	Solution(s)
Error logging on: -2147221233 [Collaboration Data Objects - [MAPI_E_NOT_FOUND(8004010F)]]	<ul style="list-style-type: none"><li>■ Verify that CDO is installed as described in <a href="#">CDO 1.21</a> on page 5-15.</li><li>■ If CDO was installed from Exchange, install Outlook on the Central Data Collectors server (if not already installed) and install CDO from Outlook (see <a href="#">page 5-15</a>).</li></ul>
Error retrieving PR_MESSAGE_SIZE: - 2147221219 [Collaboration Data Objects - [MAPI_E_FAILONEPROVIDER(8004011D)]] <b>And</b> Error retrieving PR_CONTENT_COUNT: - 2147221219 [Collaboration Data Objects - [MAPI_E_FAILONEPROVIDER(8004011D)]]	<ul style="list-style-type: none"><li>■ Make sure that the Windows account running the Exchange Server Mailbox collector belongs to the correct group. If you set the security permissions as described in the first method in article 262054, the account must be in the Exchange Services group.</li><li>■ The Windows account <i>cannot</i> be a member of the Administrators, Domain Admins, or Enterprise Admins group.</li></ul>

---



## Identifiers and Resources Collected from the Exchange Server Mailbox Store

By default, the following object attributes in the Exchange Server mailbox store are defined as chargeback identifiers and resources in the MExchangeMbx.wsf conversion script.

For attributes defined as identifiers, a flag of `True` specifies that identifier is included as in the CSR file. A flag of `False` specifies that the identifier is not included. If you wish to exclude or include an identifier, set the flag appropriately.

The rate codes assigned to the attributes defined as resources are pre-loaded in the CIMSRate table.

Mailbox Object Attribute	Identifier Name or Resource Description in CIMS Server	Default Flag Value	Assigned Rate Code in CIMS Server
<b>Identifiers</b>			
—	Feed (defined in the Exchange Server Mailbox collector job file)	True	—
First Name	First_Name	True	—
Display Name	Display_Name	True	—
Initials	Initials	True	—
Last Name	Last_Name	True	—
Mailbox User ID	Mail_Nickname	True	—
Street Address	Street_Address	False	—
City	City	False	—
State	State	False	—
Country	Country	False	—
Title	Title	False	—
Company	Company	False	—
Department	Department	True	—
Office	Office	False	—
Assistant	Assistant	False	—
Phone Number	Phone_Number	False	—

**Table 5-7 • Default Exchange Server Mailbox Identifiers and Resources**

<b>Mailbox Object Attribute</b>	<b>Identifier Name or Resource Description in CIMS Server</b>	<b>Default Flag Value</b>	<b>Assigned Rate Code in CIMS Server</b>
Email Address	E_MailAddress	True	—
Employee Number	Employee_Number	False	—
Employee Type	Employee_Type	False	—
Extension Attribute 1	Extension_Attribute_1	True	—
Extension Attribute 2	Extension_Attribute_2	True	—
Extension Attribute 3	Extension_Attribute_3	True	—
Extension Attribute 4	Extension_Attribute_4	True	—
Extension Attribute 5	Extension_Attribute_5	True	—
Extension Attribute 6	Extension_Attribute_6	False	—
Extension Attribute 7	Extension_Attribute_7	False	—
Extension Attribute 8	Extension_Attribute_8	False	—
Extension Attribute 9	Extension_Attribute_9	False	—
Extension Attribute 10	Extension_Attribute_10	False	—
Extension Attribute 11	Extension_Attribute_11	False	—
Extension Attribute 12	Extension_Attribute_12	False	—
Extension Attribute 13	Extension_Attribute_13	False	—
Extension Attribute 14	Extension_Attribute_14	False	—
Extension Attribute 15	Extension_Attribute_15	False	—

**Table 5-7 • Default Exchange Server Mailbox Identifiers and Resources (Continued)**

Mailbox Object Attribute	Identifier Name or Resource Description in CIMS Server	Default Flag Value	Assigned Rate Code in CIMS Server
<b>Resources</b>			
—	MS Exchange Mailbox Count (Mailbox days)  <b>Note:</b> Because the collector is gathering data from a specific mailbox, the value for this is always 1.	—	EXMBXCNT
PR_MESSAGE_SIZE	MS Exchange Mailbox Size (MB days)  <b>Note:</b> To collect this resource, you need to specify <code>CollectMailBoxSize="true"</code> in the Exchange Server Mailbox job file. See <a href="#">page 5-22</a> .	—	EXMBXSIZ
PR_CONTENT_COUNT	MS Exchange Mailbox Messages (Message days)  <b>Note:</b> To collect this resource, you need to specify <code>CollectMailBoxSize="true"</code> in the Exchange Server Mailbox job file. See <a href="#">page 5-22</a> .	—	EXMBXMSG

Table 5-7 • Default Exchange Server Mailbox Identifiers and Resources (Continued)

## Setting Up the Exchange Server Mailbox Collector

On the central CIMS Data Collectors server, set up an XML job file for the Exchange Server Mailbox collector as described in see [Creating Job Files](#) on page 2-25. The following is an example process for the collector in the job file:

```
<Process id="MSExchangeMbx"
description="Process for Exchange Server Mailbox Collector"
joblogShowStepOutput="true"
joblogShowStepParameters="true"
active="true">
  <Defaults>
    <Default LogDate="RNDATE"/>
  </Defaults>
  <Steps>
    <Step id="Server1 Collection"
description="Server1 MSExchangeMbx"
type="ConvertToCSR"
programName="MSExchange\Mailbox\MSExchangeMbx.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter ServerName="DC=ABCISOFT,DC=Corp"/>
        <Parameter Organization="ORGNAME"/>
        <Parameter Site="SITENAME"/>
        <Parameter CollectMailBoxSize="true"/>
        <Parameter AllowErrorOnMailBoxSizeResources="true"/>
        <Parameter IdentifierName=""/>
        <Parameter IdentifierValue=""/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan MSExchangeMbx"
type="Process"
programName="Scan"
programType="net"
active="true">
      </Step>
    <Step id="Process"
description="Standard Processing for MSExchangeMbx"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
      </Step>
    <Step id="DatabaseLoad"
description="Database Load for MSExchangeMbx"
type="Process"
programName="DBLoad"
programType="com"
active="true">
      </Step>
  </Steps>
</Process>
```

```

    <Step id="Cleanup"
        description="Cleanup MExchangeMbx"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Exchange Server Mailbox collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 5-8](#). These parameters are used by the conversion script, `MExchangeMbx.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The Exchange Server Mailbox collector collects data that is current as of the date and time that the collector is run by CIMS Job Runner. However, the start and end date that appear in the CSR file records and the date that appears in the initial CSR file name will reflect the value entered for the <code>LogDate</code> parameter. For example, if you use the <code>LogDate</code> parameter <code>PREDAY</code>, the previous day's date is used.</p> <p>To include the actual date that the data was collected, you need to include the parameter <code>LogDate="RNDATE"</code> at the job or process level in the job file (see the example on <a href="#">page 5-20</a>).</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server running Exchange Server.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR files that are created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>

**Table 5-8 • MExchangeMbx.wsf Parameters**

Parameter	Description/Values
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id</code> = "MSExchangMbx", the output folder is MSExchangeMbx.</p>
ServerName	<p>For Exchange Server 5.5, the name of the server running Exchange Server.</p> <p>For Exchange Server 2000 and 2003, the Active Directory distinguished name (<i>DN</i>) that you want to search for Exchange data. Usually, the value will be the DN of the Active Directory root object.</p>
Organization	<p>For Exchange Server 5.5, the organization unit name. For example, ABC, for the ABC organization.</p> <p>For Exchange Server 2000 and 2003, use the value ORGNAME.</p>
Site	<p>For Exchange Server 5.5, the site name. For example, ABCSF, for the ABC organization San Francisco site.</p> <p>For Exchange Server 2000 and 2003, use the value SITENAME.</p>
CollectMailboxSize (optional)	<p>If this parameter is set to "true", the mailbox size and number of messages resources are collected.</p> <p>If the parameter is not included, is left blank, or is set to "false", these resources are not collected.</p> <p><b>Note:</b> You need to install the CDO 1.21 library to collect these resources. See <i>CDO 1.21</i> on page 5-15.</p>

**Table 5-8 • MSExchangeMbx.wsf Parameters (Continued)**

Parameter	Description/Values
AllowErrorOnMailBoxSize Resources (optional)	<p>This parameter is applicable only if the CollectMailboxSize parameter is set to "true".</p> <p>If this parameter is set to "true", processing will continue if an error is encountered when collecting the size and number of messages for a mailbox.</p> <p>If this parameter is not included, is left blank, or is set to "false", processing fails for all mailboxes.</p>
IdentifierName and IdentifierValue (optional)	<p>This parameter enables you to filter data that you want to collect from the Exchange Server mailbox store by a specific object attribute name and value. For example, if you are using the Department attribute to store each user's department and would like to retrieve data for department 8091 only, you would enter the following in the job file:</p> <pre>"identifierName="Department" "identifierValue="8091"</pre>

**Table 5-8 • MExchangeMbx.wsf Parameters (Continued)**

## Running the Exchange Server Mailbox Collector

Use CIMS Job Runner to run the Exchange Server Mailbox collector as described in *Running CIMS Job Runner* on page 2-93.

## **Microsoft Outlook Web Access Data Collection**

Outlook Web Access enables you to read and send e-mail messages via a Web interface directly to Exchange Server. Therefore, the metrics provided by the standard IIS logs provide information about Web access of the Outlook Web Access page. The metrics provided by the standard Exchange Server logs capture the e-mail messages sent and received.

For information about Exchange Server log files, see *Microsoft Exchange Server 5.5 Collector* on page 5-2 or *Microsoft Exchange Server 2000/2003 Collectors* on page 5-6. For information about IIS log files, see *Microsoft Internet Information Services (IIS) Data Collector* on page 6-3.

## **Lotus Notes Data Collector**

CIMS Lab provides a CIMS Data Collector for Lotus Notes. For instructions on how to configure this collector, contact CIMS Lab.



# Internet Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for Internet applications. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Microsoft Internet Information Services (IIS)</b>	
<b>Data Collector</b> .....	<b>6-3</b>
Enabling IIS Logging .....	6-3
IIS Log File Format .....	6-4
Identifiers and Resources Collected from the IIS Log File .....	6-6
Setting Up the IIS Collector .....	6-8
Running the IIS Collector .....	6-10
<b>Microsoft Internet Security and Acceleration (ISA) Server Data Collector</b> .....	<b>6-11</b>
Enabling ISA Logging .....	6-11
ISA Server Log File Format .....	6-12
Identifiers and Resources Collected from the ISA Server Log File .....	6-16
Setting Up the ISA Server Collector .....	6-17
Running the ISA Server Collector .....	6-18
<b>Microsoft Proxy Server Data Collector</b> .....	<b>6-19</b>
Enabling Proxy Server Logging .....	6-19
Proxy Server Log File Format .....	6-20
Identifiers and Resources Collected from the Proxy Server Log File .....	6-24
Setting Up the Proxy Server Collector .....	6-25
Running the Proxy Server Collector .....	6-26
<b>SQUID Data Collector</b> .....	<b>6-27</b>
Identifiers and Resources Collected from the	
SQUID Log File .....	6-27
Setting Up the SQUID Collector .....	6-27
Running the SQUID Collector .....	6-29

<b>Sendmail Data Collector</b> .....	<b>6-30</b>
Identifiers and Resources Collected from the Sendmail Log File .....	6-30
Setting Up the Sendmail Collector .....	6-30
Running the Sendmail Collector .....	6-32
<b>Apache Data Collector</b> .....	<b>6-33</b>
Identifiers and Resources Collected from the Apache Log File .....	6-33
Setting Up the Apache Collector .....	6-33
Running the Apache Collector .....	6-35
<b>Netscape Proxy Server Data Collector</b> .....	<b>6-35</b>

## Microsoft Internet Information Services (IIS) Data Collector

The CIMS Data Collector for Microsoft IIS collects data that is contained in a log file produced by IIS. This log file provides useful metrics such as:

- Bytes sent from a client IP address to a server IP address.
- Bytes sent from a server IP address to a client IP address.
- User name and IP address, IIS site name, and server name and IP address.

You can process log files for IIS Web and FTP sites and the SMTP server.

The following sections provide instructions for enabling logging for IIS and for setting up and running the IIS collector.

### Enabling IIS Logging

The following are instructions for enabling logging for IIS 5.1 on the Windows 2000 Server operating system. Refer to the Microsoft documentation for instructions on how to enable logging on other platforms. You need to enable logging for each Web and FTP site and SMTP server individually.

- 1 In the Windows Internet Information Services window, right-click the site or server, and then click **Properties**.

The properties dialog box appears.

- 2 On the **Web Site** tab, select **Enable Logging** and click **W3C Extended Log File Format** from the **Active log format** list.

- 3 Click **Apply**, and then click **Properties**.

The Extended Logging Properties dialog box appears.

- 4 On the **General Properties** tab, set the general properties such as the log schedule (daily, weekly, monthly, etc.) and log file location.
- 5 On the **Extended Properties** tab, select the properties that you want to log. CIMS Lab recommends that you select all **Extended Properties**. You can also select **Process Accounting** properties; however, this information is not useful for chargeback and is not written to the CIMS Server Resource (CSR) file.
- 6 Click **OK** to save the settings and close the dialog box, and then click **OK** again to close the properties dialog box.

For more information about IIS logging, refer to the Microsoft documentation.

## IIS Log File Format

The following table describes the record fields in the IIS log file.

Field Name	Description/Values
date	The date that the action occurred.
time	The time that the action occurred.
c-ip (client IP address)	The IP address of the client that accessed the server.
cs-username (user name)	The name of the authenticated user who accessed the server. This does not include anonymous users, which are represented by a hyphen (-).
s-sitename (service name)	The Internet service and instance number that was accessed by the client.
s-computername (server name)	The name of the server on which the log entry was generated.
s-ip (server IP address)	The IP address of the server on which the log entry was generated.
s-port (server port)	The port number the client was connected to.
cs-method (method)	The action the client was trying to perform (for example, a GET method).
cs-uri-stem (URI stem)	The resource accessed (for example, Default.htm).
cs-uri-query (URI query)	The query, if any, the client was trying to perform.
sc-status (protocol status)	The status of the action, in HTTP or FTP terms.
sc-win32-status (protocol status)	The status of the action, in terms used by Windows.
sc-bytes (bytes sent)	The number of bytes sent by the server.
cs-bytes (bytes received)	The number of bytes received by the server.
time-taken	The length of time the action took.
cs-version (protocol version)	The protocol (HTTP, FTP) version used by the client. For HTTP, this will be either HTTP 1.0 or HTTP 1.1.
cs-host (host)	Displays the content of the host header.

**Table 6-1 • IIS Log File Format**

Field Name	Description/Values
cs(User-Agent) (user agent)	The browser used on the client.
cs(Cookie) (cookie)	The content of the cookie sent or received, if any.
cs(Referer)	The previous site visited by the user. This site provided a link to the current site.

**Table 6-1 • IIS Log File Format (Continued)**

## Identifiers and Resources Collected from the IIS Log File

By default, the following fields in the IIS log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `MSIIS.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>		
—	Feed (defined in the IIS collector job file)	—
	c_ip	—
	c_ip0	—
c-ip	c_ip1	—
	c_ip2	—
	c_ip3	—
cs-username	User	—
s-sitename	sitename	—
s-computername	Server	—
	s_ip	—
	s_ip0	—
s-ip	s_ip1	—
	s_ip2	—
	s_ip3	—
s-port	s-port	—
cs-uri-stem	TOPDIR	—
cs-uri-stem	NEXTDIR	—
cs-host	cs-host	—

**Table 6-2 • Default IIS Identifiers and Resources**

Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>FTP Resources</b>		
cs-bytes	IIS FTP Bytes Received	FCSBytes
sc-bytes	IIS FTP Bytes Sent	FSCBytes
sc-status	IIS FTP Successful Protocol Status 2xx	FIIS-2
sc-status	IIS FTP Redirection Protocol Status 3xx	FIIS-3
sc-status	IIS FTP Client Error Protocol Status 4xx	FIIS-4
sc-status	IIS FTP Server Error Protocol Status 5xx	FIIS-5
time-taken	IIS FTP Time Taken	FTimeTkn
<b>SMTP Resources</b>		
cs-bytes	IIS SMTP Bytes Received	SCSBytes
sc-bytes	IIS SMTP Bytes Sent	SSCBytes
sc-status	IIS SMTP Successful Protocol Status 2xx	SIIS-2
sc-status	IIS SMTP Redirection Protocol Status 3xx	SIIS-3
sc-status	IIS SMTP Client Error Protocol Status 4xx	SIIS-4
sc-status	IIS SMTP Server Error Protocol Status 5xx	SIIS-5
time-taken	IIS SMTP Time Taken	STimeTkn
<b>Web Resources</b>		
cs-bytes	IIS Web Bytes Received	WCSBytes
sc-bytes	IIS Web Bytes Sent	WSCBytes
sc-status	IIS Web Successful Protocol Status 2xx	WIIS-2
sc-status	IIS Web Redirection Protocol Status 3xx	WIIS-3
sc-status	IIS Web Client Error Protocol Status 4xx	WIIS-4
sc-status	IIS Web Server Error Protocol Status 5xx	WIIS-5
time-taken	IIS Web Time Taken	WTimeTkn

**Table 6-2 • Default IIS Identifiers and Resources (Continued)**

## Setting Up the IIS Collector

On the central CIMS Data Collectors server, set up an XML job file for the IIS collector as described in *Creating Job Files* on page 2-25. The following is an example process for the IIS collector in the job file. This process collects IIS data for all web sites on Server1 and Server2.

```
<Process id="MSIIS-Web"
description="Process for IIS Collector"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 IIS"
type="ConvertToCSR"
programName="MSIIS\MSIIS.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
        <Parameter ProcessType="web"/>
        <Parameter SiteIDOrAll="All"/>
      </Parameters>
    </Step>
    <Step id="Server2 Collection"
description="Server2 IIS"
type="ConvertToCSR"
programName="MSIIS\MSIIS.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server2"/>
        <Parameter LogFolder="\\Server2\LogFiles"/>
        <Parameter ProcessType="web"/>
        <Parameter SiteIDOrAll="All"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan MSIIS"
type="Process"
programName="Scan"
programType="net"
active="true">
    </Step>
    <Step id="Process"
description="Standard Processing for MSIIS"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
    <Step id="DatabaseLoad"
description="Database Load for MSIIS"
type="Process"
programName="DBLoad"
programType="com"
active="true">
    </Step>
  </Steps>
```



```

        <Step id="Cleanup"
            description="Cleanup MSIIIS"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the IIS collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 6-3](#). These parameters are used by the conversion script, `MSIIIS.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	The name of the server that contains the log file that you want to collect.  A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.  This parameter is included as an identifier in the CSR file.
OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.  The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id</code> ="MSIIIS-Web", the output folder is <code>MSIIIS-Web</code> .
LogFolder	The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.

**Table 6-3 • MSIIIS.wsf Parameters**

## ■ Internet Data Collectors

### Microsoft Internet Information Services (IIS) Data Collector

---

Parameter	Description/Values
ProcessType	The IIS processing Type: Web, FTP, or SMTP.
SiteIDOrAll (optional)	This parameter specifies data collection from a particular IIS site or from all IIS sites on a server. Valid values are a particular site or All. If this parameter is not included or is left blank, the default is All.

---

**Table 6-3 • MSIS.wsf Parameters (Continued)**

## Running the IIS Collector

Use CIMS Job Runner to run the IIS collector as described in *Running CIMS Job Runner* on page 2-93.

## Microsoft Internet Security and Acceleration (ISA) Server Data Collector

The CIMS Data Collector for Microsoft ISA Server collects data that is contained in a log files produced by ISA Server. This log file provides useful metrics such as the number of bytes received from and sent to a remote computer and the total time taken to process a request.

The following sections provide instructions for enabling logging for ISA Server and for setting up and running the ISA Server collector.

### Enabling ISA Logging

The following provides an example of enabling logging for ISA Server 2000:

#### *To configure logging to a file:*

- 1 In the ISA Management window, click **Logs**.
- 2 In the details pane, right-click the applicable service, and then click **Properties**.
- 3 On the **Log** tab, click **File**.
- 4 Click **Options**.
- 5 Select the **Compress log files** check box.

#### *To specify fields to log:*

- 1 In the ISA Management window, click **Logs**.
- 2 On the details pane, right-click the applicable service, and then click **Properties**.
- 3 On the **Fields** tab, do one of the following:
  - To select specific fields, select the appropriate check box.
  - To clear all of the check boxes in the field list, click **Clear All**.
  - To select all of the check boxes in the field list, click **Select All**.
  - To select a default set of fields in the ISA Server log file, click **Restore Defaults**.

For more information about ISA logging, refer to the Microsoft documentation, including the Microsoft Knowledge Base Article 302372 (<http://support.microsoft.com/default.aspx?scid=kb;en-us;302372>).

## ISA Server Log File Format

The fields that appear in the ISA Server log file depend on the fields selected when configuring logging (see *To specify fields to log:* on page 6-11). The following table describes all of the possible record fields in the ISA Server log file. The field names (noted in parentheses) appear when you use the W3C extended log file format.

Field Name	Description/Values
Client IP (c-ip)	The IP address of the requesting client.
Client User Name (cs-username)	The Windows 2000 logon account name of the user making the request. If ISA Server Access Control is not being used, ISA Server uses <code>anonymous</code> .
Client User Agent (c-agent)	The client application type sent by the client in the HTTP header. When ISA Server is actively caching, the client agent is <code>ISA Server</code> .  For the Firewall service, this field includes information about the client's operating system.
Authentication Status (sc-authenticated)	Indicates whether or not client has been authenticated with ISA Server. Possible values are <code>Y</code> and <code>N</code> .
Log Date (date)	The date that the logged event occurred.
Log Time (time)	The time that the logged event occurred.
Service Name (s-svcname)	The name of the service that is logged: <ul style="list-style-type: none"> <li>■ <code>w3proxy</code> indicates outgoing Web requests to the Web Proxy service.</li> <li>■ <code>fwsrv</code> indicates Firewall service.</li> <li>■ <code>w3reverseproxy</code> indicates incoming Web requests to the Web Proxy service.</li> </ul>
Proxy Name (s-computername)	The name of the computer running ISA Server. This is the computer name that is assigned in Windows 2000.
Referring Server Name (cs-referred)	If ISA Server is used upstream in a chained configuration, this indicates the server name of the downstream server that sent the request.
Destination Name (r-host)	The domain name for the remote computer that provides service to the current connection. For the Web Proxy service, a hyphen (-) in this field may indicate that an object was retrieved from the Web Proxy server cache and not from the destination.

**Table 6-4 • ISA Server Log File Format**

Field Name	Description/Values
Destination IP (r-ip)	The network IP address for the remote computer that provides service to the current connection. For the Web Proxy service, a hyphen (-) in this field may indicate that an object was sourced from the Web Proxy server cache and not from the destination. One exception is negative caching. In that case, this field indicates a destination IP address for which a negative-cached object was returned.
Destination Port (r-port)	The reserved port number on the remote computer that provides service to the current connection. This is used by the client application initiating the request.
Processing Time (time-taken)	<p>This indicates the total time, in milliseconds, that is needed by ISA Server to process the current connection. It measures elapsed server time from the time that the server first received the request to the time when final processing occurred on the server—when results were returned to the client and the connection was closed.</p> <p>For cache requests that were processed through the Web Proxy service, processing time measures the elapsed server time needed to fully process a client request and return an object from the server's cache to the client.</p>
Bytes Sent (cs-bytes)	The number of bytes sent from the client to the server during the current connection. A hyphen (-) or a zero or negative number in this field indicates that this information was not provided by the remote computer, or that no bytes were sent to the remote computer.
Bytes Received (sc-bytes)	The number of bytes sent from the server and received by the client during the current connection. A hyphen (-) or a zero or negative number in this field indicates that this information was not provided by the remote computer, or that no bytes were received from the server.
Protocol Name (cs-protocol)	<p>Specifies the application protocol used for the connection. Common values are HTTP, FTP, Gopher, and HTTPS (Secure Hypertext Transfer Protocol).</p> <p>For Firewall service, the port number is also logged.</p>
Transport (cs-transport)	Specifies the transport protocol used for the connection. Common values are TCP and UDP.

---

**Table 6-4 • ISA Server Log File Format (Continued)**

Field Name	Description/Values
Operation (s-operation)	<p>Specifies the application method used.</p> <p>For the Web Proxy service, common values are GET, PUT, POST, and HEAD.</p> <p>For the Firewall service, common values are CONNECT, BIND, SEND, RECEIVE, GHBN (GetHostByName), and GHBA (GetHostByAddress).</p>
Object Name (cs-uri)	<p>For the Web Proxy service, this field shows the contents of the URL request. This field applies only to the Web Proxy service log.</p>
Object MIME (cs-mime-type)	<p>The Multipurpose Internet Mail Extensions (MIME) type for the current object. This field may also contain a hyphen (-) to indicate that this field is not used or that a valid MIME type was not defined or supported by the remote computer.</p> <p>This field applies only to the Web Proxy service log.</p>
Object Source (s-object-source)	<p>Indicates the source that was used to retrieve the current object. This field applies only to the Web Proxy service log.</p>
Result Code (sc-status)	<p>This field can be used to indicate:</p> <ul style="list-style-type: none"> <li>■ For values less than 100, a Windows (Win32) error code.</li> <li>■ For values between 100 and 1,000, an HTTP status code.</li> <li>■ For values between 10,000 and 11,004, a Winsock error code.</li> </ul>
Cache Info (s-cache-info)	<p>This number reflects the cache status of the object, which indicates why the object was or was not cached.</p> <p>This field applies only to the Web Proxy service log.</p>
Rule #1 (rule#1)	<p>The rule that either allowed or denied access to the request.</p>
Rule #2 (rule#2)	<p>The second rule that either allowed or denied access to the request.</p>

**Table 6-4 • ISA Server Log File Format (Continued)**

Field Name	Description/Values
Session ID ( <code>sessionid</code> )	<p>Identifies a session's connections.</p> <p>For Firewall clients, each process that connects through the Firewall service initiates a session.</p> <p>For secure network address translation (SecureNAT) clients, a single session is opened for all the connections that originate from the same IP address.</p> <p>This field applies only to the Firewall service log.</p>
Connection ID ( <code>connectionid</code> )	<p>Identifies entries that belong to the same socket. Outbound TCP usually has two entries for each connection: when the connection is established and when the connection is terminated. UDP usually has two entries for each remote address.</p> <p>This field applies only to the Firewall service log.</p>

---

**Table 6-4 • ISA Server Log File Format (Continued)**

## Identifiers and Resources Collected from the ISA Server Log File

By default, the following fields in the ISA Server log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `MSISA.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>		
—	Feed (defined in the ISA Server Mailbox collector job file)	—
Client User Name (cs-username)	User	—
<b>Resources</b>		
Processing Time (time-taken)	MS ISA Server Time Taken	ISATIME
Bytes Sent (cs-bytes)	MS ISA Server Bytes Sent	ISASENT
Bytes Received (sc-bytes)	MS ISA Server Bytes Received	ISARECV

**Table 6-5 • Default ISA Server Identifiers and Resources**



## Setting Up the ISA Server Collector

On the central CIMS Data Collectors server, set up an XML job file for the ISA Server collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file:

```

<Process id="MSISA"
  description="Process for ISA Server Collector"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 ISA"
      type="ConvertToCSR"
      programName="MSISA\MSISA.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan MSIISA"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for MSISA"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for MSISA"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup MSISA"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the ISA Server collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 6-6](#). These parameters are used by the conversion script, `MSISA.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
<code>LogDate</code>	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
<code>RetentionFlag</code>	This parameter is for future use.
<code>Feed</code>	The name of the server that contains the log file that you want to collect.  A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.  This parameter is included as an identifier in the CSR file.
<code>OutputFolder</code>	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.  The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id</code> = "MSISA", the output folder is MSISA.
<code>LogFolder</code>	The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.

**Table 6-6 • MSISA.wsf Parameters**

## Running the ISA Server Collector

Use CIMS Job Runner to run the ISA Server collector as described in [Running CIMS Job Runner](#) on page 2-93.

## Microsoft Proxy Server Data Collector

The CIMS Data Collector for Microsoft Proxy Server collects data that is contained in a log files produced by Proxy Server. This log file provides useful metrics such as the number of bytes received from and sent to a remote computer and the total time taken to process a request.

The following sections provide instructions for enabling logging for Proxy Server and for setting up and running the Proxy Server collector.

### Enabling Proxy Server Logging

The following are instructions for enabling logging for the Proxy Server 2.0 Web Proxy service on the Windows NT Server 4.0 operating system. You can also follow these instructions to enable logging for the WinSock Proxy and Socks Proxy services. Refer to the Microsoft documentation for instructions on how to enable logging on other platforms.

- 1 In the Microsoft Management Console window, right-click **Web Proxy**.
- 2 Click **Properties**. The Web Proxy Service Properties dialog box appears.
- 3 On the **Logging** tab, select the **Enable logging using** check box and set the general properties such as the log format (regular or verbose), schedule (daily, weekly, monthly, etc.), and location.
- 4 On the **Permissions** tab, select the **Enable access control** check box and click **WWW** in the **Protocol** list. Enabling access control causes the NT User ID to be added to each record of the log file.
- 5 Click **OK** to save the settings and close the dialog box.

For more information about Proxy Server logging, refer to the Microsoft documentation.

## Proxy Server Log File Format

The following table describes the record fields in the Proxy Server log file.

Field Name	Description/Values	Regular Logging	Verbose Logging
Client IP	The IP address of the requesting client. In cases where active caching is occurring, this field is the same as the Proxy Name field.	X	X
Client User Name	The Windows NT logon account name of the user making the request.	X	X
Client Agent	For the Web Proxy service, indicates specialized header information from the client browser to use when processing the proxy request.  For the WinSock Proxy service, indicates the name of the client application that is generating the Windows Socket process request.		X
Client Platform	For the Web Proxy service, this field is not used.  For the WinSock Proxy service, indicates the client operating system.		X
Authentication Status	Indicates whether or not the service request is using an authenticated client connection to Proxy Server. Possible values are Y and N.		X
Log Date	The date that the logged event occurred.	X	X
Log Time	The time that the logged event occurred.	X	X
Service Name	The name of the active service being logged:  ■ CERNProxy indicates Web Proxy service logging.  ■ WSPProxy indicates WinSock Proxy service logging.  ■ SOCKS indicates Socks Proxy service logging.	X	X

**Table 6-7 • Proxy Server Log File Format**

Field Name	Description/Values	Regular Logging	Verbose Logging
Proxy Name	The name of the computer running Proxy Server. This is the name assigned in Windows NT Server 4.0 for the computer.		X
Referring Server Name	If Proxy Server is used upstream in a chained configuration, this indicates the server name of the downstream server that sent the request.		X
Destination Name	The domain name for the remote computer that provides service to the current connection. For the Web Proxy service, a hyphen (-) in this field may indicate that an object was sourced from the Web Proxy Server cache and not from the destination. One exception is negative caching. In that case, this field indicates a destination name for which a negative cached object was returned.	X	X
Destination IP	The network IP address for the remote computer that provides service to the current connection. For the Web Proxy service, a hyphen (-) in this field may indicate that an object was sourced from the Web Proxy Server cache and not from the destination. One exception is negative caching. In that case, this field indicates a destination IP address for which a negative cached object was returned.		X
Destination Port	The reserved port number on the remote computer that provides service to the current connection. This is used by the client application initiating the request.	X	X

---

**Table 6-7 • Proxy Server Log File Format (Continued)**

Field Name	Description/Values	Regular Logging	Verbose Logging
Processing Time	<p>This indicates the total time, in milliseconds, that is needed by Proxy Server to process the current connection. It measures elapsed server time from when the server first received the request to the time when final processing occurred on the server—when results were returned to the client and the connection was closed.</p> <p>For cache requests processed through the Web Proxy service, processing time measures the elapsed server time needed to fully process a client request and return an object from the server’s cache to the client.</p> <p>For the WinSock Proxy service, the number of bytes received when a connection is terminated. This is in addition to the log generated when the request is processed.</p>		X
Bytes Sent	<p>For the Web Proxy service, the number of bytes sent from the client to the server during the current connection. A hyphen (-) or a zero or negative number in this field indicates that this information was not provided by the remote computer, or that no bytes were sent to the remote computer.</p> <p>For the WinSock Proxy service, the number of bytes sent when a connection is terminated. This is in addition to the log generated when the request is processed.</p>		X
Bytes Received	<p>For the Web Proxy service, the number of bytes sent from the server and received by the client during the current connection. A hyphen (-) or a zero or negative number in this field indicates that this information was not provided by the remote computer, or that no bytes were received from the server.</p>		X

**Table 6-7 • Proxy Server Log File Format (Continued)**

Field Name	Description/Values	Regular Logging	Verbose Logging
Protocol Name	For the Web Proxy service, specifies the protocol used for transfer (such as HTTP, FTP, or Gopher).  For the WinSock Proxy service, specifies a well-known destination port number for the socketed application (such as port 1070 for RealAudio).	X	X
Transport	For the Web Proxy Service, this is always TCP/IP.  For the WinSock Proxy service, this can be TCP/IP, UDP, or IPX/SPX.		X
Operation	For the Web Proxy service, specifies the current HTTP method used. Possible values are GET, PUT, POST, and HEAD.  For the WinSock Proxy service, specifies the current socket API call in use. Possible values include Connect( ), Accept( ), SendTo( ), RecvFrom( ), GetHostByName( ), and Listen( ).		X
Object Name (Uri)	For the Web Proxy service, this field shows the contents of the URL request.	X	X
Object MIME	For the Web Proxy service, the Multipurpose Internet Mail Extensions (MIME) type for the current object. This field may also be contain a hyphen (-) to indicate that this field is not used, or that a valid MIME type was not defined or supported by the remote computer.		X

Table 6-7 • Proxy Server Log File Format (Continued)

Field Name	Description/Values	Regular Logging	Verbose Logging
Object Source	For the Web Proxy service, indicates the source used to retrieve the current object.	X	X
Result Code	For the Web Proxy service, this field can be used to indicate: <ul style="list-style-type: none"> <li>■ For values less than 100, a Windows (Win32) error code.</li> <li>■ For values between 100 and 1,000, an HTTP status code.</li> <li>■ For values between 10,000 and 11,004, a Winsock error code.</li> </ul>	X	X

**Table 6-7 • Proxy Server Log File Format (Continued)**

## Identifiers and Resources Collected from the Proxy Server Log File

By default, the following fields in the Proxy Server log file are defined as chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `MSProxy.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

Log File Field	Identifier Name	Rate Code
<b>Identifiers</b>		
—	Feed (defined in the Proxy Server collector job file)	—
Client User Name (cs-username)	User	—
<b>Resources</b>		
Processing Time	—	PROXTIME
Bytes Sent	—	PROXSENT
Bytes Received	—	PROXRECV

**Table 6-8 • Default Proxy Server Identifiers and Resources**



## Setting Up the Proxy Server Collector

On the central CIMS Data Collectors server, set up an XML job file for the Proxy Server collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file:

```

<Process id="MSProxy"
  description="Process for Proxy Server Collector"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 MSProxy"
      type="ConvertToCSR"
      programName="MSProxy\MSProxy.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan MSProxy"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for MSProxy"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for MSProxy"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup MSProxy"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

For a description of the Parameter element attributes that are specific to the Proxy Server collector (that is, the parameters provided for the ConvertToCSR step), see [Table 6-9](#). These parameters are used by the conversion script, MSProxy.wsf.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	The name of the server that contains the log file that you want to collect.  A subfolder with the same name as the server is automatically created in the process definition folder (see the OutputFolder parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.  This parameter is included as an identifier in the CSR file.
OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.  The output folder is defined by the Process id attribute in the job file. For example, if the Process id="MSProxy", the output folder is MSProxy.
LogFolder	The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.

**Table 6-9 • MSProxy.wsf Parameters**

## Running the Proxy Server Collector

Use CIMS Job Runner to run the Proxy Server collector as described in [Running CIMS Job Runner](#) on page 2-93.

## SQUID Data Collector

The CIMS Data Collector for SQUID collects data that is contained in a log file produced by SQUID. This log file provides useful metrics such as the number of bytes received from and sent to a remote computer and the total time taken to process a request.

### Identifiers and Resources Collected from the SQUID Log File

By default, the following field values in the log file are defined as chargeback identifiers and resource rate codes (see the `DefineIdentifier` and `DefineResource` methods in the `SQUID.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

#### Identifiers

- Feed (defined in the SQUID collector job file)
- ClientIP

#### Resource Rate Codes

- SQUIDSNT (bytes sent)
- SQUIDRCV (bytes received)

### Setting Up the SQUID Collector

On the central CIMS Data Collectors server, set up an XML job file for the SQUID collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file:

```
<Process id="SQUID"
  description="Process for SQUID Collector"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 SQUID"
      type="ConvertToCSR"
      programName="Squid\squid.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
```

```

        <Step id="Scan"
            description="Scan SQUID"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
        </Step>
        <Step id="Process"
            description="Standard Processing for SQUID"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">

        <Step id="DatabaseLoad"
            description="Database Load for SQUID"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>

        <Step id="Cleanup"
            description="Cleanup SQUID"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the SQUID collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 6-10](#). These parameters are used by the conversion script, `squid.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.

**Table 6-10 • SQUID.wsf Parameters**

Parameter	Description/Values
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="SQUID"</code>, the output folder is <code>SQUID</code>.</p>
LogFolder	<p>The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.</p>

**Table 6-10 • SQUID.wsf Parameters (Continued)**

## Running the SQUID Collector

Use CIMS Job Runner to run the SQUID collector as described in *Running CIMS Job Runner* on page 2-93.

## Sendmail Data Collector

The CIMS Data Collector for sendmail collects data that is contained in a log file produced by sendmail. This log file provides useful metrics such as the number of e-mail messages and bytes received from and sent to a remote computer.

### Identifiers and Resources Collected from the Sendmail Log File

By default, the following field values in the log file are defined as chargeback identifiers and resource rate codes (see the `DefineIdentifier` and `DefineResource` methods in the `sendmail.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

#### Identifiers

- Feed (defined in the sendmail collector job file)
- Email

#### Resource Rate Codes

- SMEMRCV (e-mail messages received)
- SMBYRCV (bytes received)
- SMEMSNT (e-mail messages sent)
- SMBYSNT (bytes sent)

### Setting Up the Sendmail Collector

On the central CIMS Data Collectors server, set up an XML job file for the sendmail collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file:

```
<Process id="SendMail"
description="Process for SendMail Collector"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 SendMail"
type="ConvertToCSR"
programName="SendMail\sendmail.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
```

```

    <Step id="Scan"
        description="Scan SendMail"
        type="Process"
        programName="Scan"
        programType="net"
        active="true">
    </Step>
    <Step id="Process"
        description="Standard Processing for SendMail"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>
    <Step id="DatabaseLoad"
        description="Database Load for SendMail"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
    <Step id="Cleanup"
        description="Cleanup SendMail"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the sendmail collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 6-11](#). These parameters are used by the conversion script, `sendmail.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.

**Table 6-11 • sendmail.wsf Parameters**

Parameter	Description/Values
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="SendMail"</code>, the output folder is <code>SendMail</code>.</p>
LogFolder	<p>The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.</p>

Table 6-11 • `sendmail.wsf` Parameters (Continued)

## Running the Sendmail Collector

Use CIMS Job Runner to run the sendmail collector as described in *Running CIMS Job Runner* on page 2-93.



## Apache Data Collector

The CIMS Data Collector for Apache collects data that is contained in a log file produced by Apache. This log file provides useful metrics such as the number of Web server hits and the number of bytes transferred from the Web server.

### Identifiers and Resources Collected from the Apache Log File

By default, the following field values in the log file are defined as chargeback identifiers and resource rate codes (see the `DefineIdentifier` and `DefineResource` methods in the `Apache.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

#### Identifiers

- Feed (defined in the Apache collector job file)
- RemoteHost
- User
- AuthUser

#### Resource Rate Codes

- APHITS (Apache total hits)
- APBYTES (bytes transferred)

### Setting Up the Apache Collector

On the central CIMS Data Collectors server, set up an XML job file for the Apache collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector:

```
<Process      id="Apache"
             description="Process for Apache Collector"
             active="true">
  <Steps>
    <Step      id="Server1 Collection"
             description="Server1 Apache"
             type="ConvertToCSR"
             programName="Apache\Apache.wsf"
             programType="wsf"
             active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
```

```

        <Step id="Scan"
            description="Scan Apache"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
        </Step>
        <Step id="Process"
            description="Standard Processing for Apache"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
        </Step>
        <Step id="DatabaseLoad"
            description="Database Load for Apache"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup Apache"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Apache collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 6-12](#). These parameters are used by the conversion script, `Apache.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the log file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.

**Table 6-12 • Apache.wsf Parameters**

Parameter	Description/Values
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="Apache"</code>, the output folder is <code>Apache</code>.</p>
LogFolder	<p>The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.</p>

**Table 6-12 • Apache.wsf Parameters (Continued)**

## Running the Apache Collector

Use CIMS Job Runner to run the Apache collector as described in *Running CIMS Job Runner* on page 2-93.

## Netscape Proxy Server Data Collector

CIMS Lab provides a CIMS Data Collector for Netscape Proxy Server. For instructions on how to configure this collector, contact CIMS Lab.

## ■ Internet Data Collectors

---

*Netscape Proxy Server Data Collector*

---

# Storage Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for disk storage. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Windows Disk Data Collector</b> .....	<b>7-2</b>
Identifiers and Resources Collected by the CIMS Windows Disk Collector .....	7-2
Setting Up the CIMS Windows Disk Collector .....	7-3
Running the CIMS Windows Disk Collector .....	7-8
<b>Veritas NetBackup</b> .....	<b>7-9</b>
NetBackup Log File Format .....	7-9
Identifiers and Resources Collected from the NetBackup Log File .....	7-10
Setting Up the NetBackup Collector .....	7-11
Running the NetBackup Collector .....	7-12
<b>Veritas</b> .....	<b>7-12</b>

## Windows Disk Data Collector

The CIMS Windows Disk collector scans a directory tree and provides a snapshot of the following:

- The amount of disk space used by each top level folder within a specified drive or folder.
- The number of files (including files in subfolders) within each of these folders.

This collector does not require a usage metering file to produce CIMS Server Resource (CSR) files. The files are produced by the collector’s executable program, CIMSWinDisk.exe. If you installed CIMS Server in the default location, this program is in C:\Program Files\CIMSLab\Collectors\CIMSWinDisk.

The following sections provide instructions for setting up and running the CIMS Windows Disk collector.

### Identifiers and Resources Collected by the CIMS Windows Disk Collector

By default, the CIMS Windows Disk collector creates the following chargeback identifiers and resource rate codes from the data collected. The rate codes are pre-loaded in the CIMSRate table.

Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>	
Feed (defined by the Feed parameter in the CIMSWinDisk.xml file [see page 7-6])	—
Folder (defined by the PathToScan parameter in the CIMSWinDisk.xml file [see page 7-7])	—
<b>Resources</b>	
MS Windows Disk Folder Usage in GB	DISKSIZE (GB days)
MS Windows Files in Folder	DISKFILE

**Table 7-1 • Default CIMS Windows Disk Identifiers and Resources**

## Setting Up the CIMS Windows Disk Collector

On the central CIMS Data Collectors server, set up an XML job file for the CIMS Windows Disk collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file. This example scans drive C of Server1 and Server2.

```
<Process id="CIMSWinDisk"
  description="Process for CIMS Windows Disk Collector"
  active="true">
  <Defaults>
    <Default LogDate="RNDATE"/>
  </Defaults>
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 CIMSWinDisk"
      type="ConvertToCSR"
      programName="CIMSWinDisk\CIMSWinDisk.exe"
      programType="console"
      active="true">
      <GenerateExternalFile filename="%ProcessFolder%\CIMSWinDisk.xml"
        overwrite="true">
        <![CDATA[
          <?xml version="1.0" encoding="utf-8" ?>
          <CIMSCollectors version = "1.0">
            <Collectors>

              <Collector name="CIMSWinDisk"
                instanceName="Server1-C"
                instanceDescription="Scan of Server1 C Drive"
                Active="True">
                <Parameters>
                  <Parameter name="LogDate" value="%RNDATE%"/>
                  <Parameter name="Retention" value="KEEP" />
                  <Parameter name="Feed" value="Server1-C" />
                  <Parameter name="OutputFolder"
                    value="%ProcessFolder%" />
                  <Parameter name="PathToScan"
                    value="\\Server1\C$" />
                  <Parameter name="Units" value="GB" />
                  <Parameter name="NumberOfLevels" value="1" />
                </Parameters>
              </Collector>
            </Collectors>
          </CIMSCollectors>
        ]]>
      </GenerateExternalFile>
      <Parameters>
        <Parameter UseStandardParameters="false"/>
        <Parameter XMLFileName="%ProcessFolder%\CIMSWinDisk.xml"/>
        <Parameter CollectorName="CIMSWinDisk"/>
      </Parameters>
    </Step>
```

```

<Step id="Server2 Collection"
description="Server2 CIMSWinDisk"
type="ConvertToCSR"
programName="CIMSWinDisk\CIMSWinDisk.exe"
programType="console"
active="true">
<GenerateExternalFile filename="%ProcessFolder%\CIMSWinDisk.xml"
overwrite="true">
  <![CDATA[
    <?xml version="1.0" encoding="utf-8" ?>
    <CIMSCollectors version = "1.0">
      <Collectors>
        <Collector name="CIMSWinDisk"
instanceName="Server2-C"
instanceDescription="Scan of Server2 C Drive"
Active="True">
          <Parameters>
            <Parameter name="LogDate" value="%RNDATE%" />
            <Parameter name="Retention" value="KEEP" />
            <Parameter name="Feed" value="Server2-C" />
            <Parameter name="OutputFolder"
value="%ProcessFolder%" />
            <Parameter name="PathToScan"
value="\\Server2\C$" />
            <Parameter name="Units" value="GB" />
            <Parameter name="NumberOfLevels" value="1" />
          </Parameters>
        </Collector>
      </Collectors>
    </CIMSCollectors>
  ]]>
</GenerateExternalFile>
<Parameters>
  <Parameter UseStandardParameters="false" />
  <Parameter XMLFileName="%ProcessFolder%\CIMSWinDisk.xml" />
  <Parameter CollectorName="CIMSWinDisk" />
</Parameters>
</Step>
<Step id="Scan"
description="Scan CIMSWinDisk"
type="Process"
programName="Scan"
programType="net"
active="true">
</Step>
<Step id="Process"
description="Standard Processing for CIMSWinDisk"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
</Step>
<Step id="DatabaseLoad"
description="Database Load for CIMSWinDisk"
type="Process"
programName="DBLoad"
programType="com"
active="true">
</Step>

```



```

    <Step id="Cleanup"
        description="Cleanup CIMSWinDisk"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

Note that the ConvertToCSR steps contain the child elements `GenerateExternalFile` and `Parameters`.

When CIMS Job Runner is run, the `GenerateExternalFile` element dynamically creates an XML file (`CIMSWinDisk.xml`) that contains parameters required by the CIMS Windows Disk collector. The parameters include the `LogDate` and `OutputFolder` parameters, which must be included in the `CIMSWinDisk.xml` file. For a description of the elements and attributes for this file, see [Defining the CIMSWinDisk.xml File](#).

---

**Note** • The file name `CIMSWinDisk.xml` is defined for the `ConvertToCSR` step for both servers. The file defined in the first step will be overwritten by the file defined in the second step. If you do not want the final `CIMSWinDisk.xml` file to appear in the process definition folder after the step has executed, use the `GenerateExternalFile` element attribute `autoRemove` (see [page 2-60](#)).

---

The `Parameters` element provides parameters for the `ConvertToCSR` steps. For a description of these parameters, see [page 2-87](#).

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

## Defining the CIMSWinDisk.xml File

The `CIMSWinDisk.xml` file contains the elements and attributes shown in [Table 7-2](#). You can generate this file automatically by using the `GenerateExternalFile` element in the job file.

Element	Attributes	Description/Values
Collector	Collector Name	The collector name. <i>Do not change this parameter.</i>
	instanceName	The name of the instance for the collector. You can assign any name that has meaning for your organization. For example, the server and drive that you are collecting from.

**Table 7-2 • CIMSWinDisk.xml Elements and Attributes**

Element	Attributes	Description/Values
Collector (continued)	instanceDescription	A description of the instance for the collector.
	Active	<p>Specifies whether the instance is included in processing. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the instance is processed)</li> <li>■ "false" (the instance is not processed)</li> </ul> <p>The default is "true".</p>
Parameter	LogDate	<p>The CIMS Windows Disk collector collects data that is current as of the date and time that the collector is run by CIMS Job Runner. However, the start and end date that appears in the output CSR file records and the date that appears in the CSR file name will reflect the value entered for this parameter. For example, if you use the LogDate parameter %PREDAY%, the previous day's date is used.</p> <p>To include the actual date that the data was collected, you need to use %RNDATE% as the LogDate parameter in the CIMSWinDisk.xml file and you need to include the parameter LogDate="RNDATE" at the job or process level in the job file (see the example on <a href="#">page 7-3</a>).</p>
	Retention	This attribute is for future use.
	Feed	<p>The name of the drive or folder that you want to collect disk space usage for.</p> <p>A subfolder with the same name as the drive/folder is automatically created in the process definition folder (see the OutputFolder parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>

**Table 7-2 • CIMSWinDisk.xml Elements and Attributes (Continued)**

Element	Attributes	Description/Values
Parameter (continued)	OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.
	PathToScan	<p>Valid values for this attribute are:</p> <ul style="list-style-type: none"> <li>■ The drive or folder one level above the folder information you want to collect. For example, "PathToScan" value="\\Server1\C\$" collects data for all top level folders under the C share.</li> </ul> <p>Note that \\Server1\C\$ is an example UNC path, which is recommended.</p> <ul style="list-style-type: none"> <li>■ All, to scan the top level folders under all drives with an administrative share (C\$ through Z\$). Note that only shared drives are scanned when you specify All.</li> </ul> <p><b>Note:</b> To scan a shared drive, the Windows user ID used to log on to the computer running the CIMS Windows Disk collector must have authority to scan the share.</p>

---

**Table 7-2 • CIMSWinDisk.xml Elements and Attributes (Continued)**

Element	Attributes	Description/Values
Parameter (continued)	Units (optional)	<p>If the attribute is set to GB, is left blank, or is not included, disk space usage is presented in gigabytes. To present the usage units in another measurement, enter one of the following values:</p> <ul style="list-style-type: none"> <li>■ bytes</li> <li>■ KB (kilobytes)</li> <li>■ MB (megabytes)</li> <li>■ A number by which you want to divide the usage units. In this case, the units are measured in bytes rather than gigabytes.</li> </ul>
	NumberOfLevels (optional)	<p>This attribute works in conjunction with the PathToScan attribute to determine the folder level that will be scanned. For example, if the PathToScan is All (scan all drives) and the NumberOfLevels attribute is 2, the data collection will reflect all second level folders under the scanned drives.</p>

**Table 7-2 • CIMSWinDisk.xml Elements and Attributes (Continued)**

### Using Smart Scan With the CIMS Windows Disk Collector

The Smart Scan feature uses the Feed parameter values to determine the files to be scanned. For most collectors, the Feed parameter is provided in the ConvertToCSR step(s) in the job file. However, the Feed parameter for the CIMS Windows Disk collector is provided in an external file. To use Smart Scan with this collector, you need to include either of the following in the ConvertToCSR step(s):

- A Feed parameter.

Or

- A scanFile="*file name*" parameter where the file name includes the full path as shown in the preceding job file example. This enables Smart Scan to determine the location of the CSR file to be scanned.

For an example of a ConvertToCSR step with this configuration and for more information about Smart Scan, see [page 2-42](#).

### Running the CIMS Windows Disk Collector

Use CIMS Job Runner to run the CIMS Windows Disk collector as described in [Running CIMS Job Runner](#) on page 2-93.

## Veritas NetBackup

**Note** • This section reflects collecting data from NetBackup on UNIX or Linux. However, CIMS Lab also supports data collection from NetBackup on Windows. For more information, contact CIMS Lab.

The CIMS Data Collector for NetBackup collects data that is contained in a log file created using the `bpimagerlist -l` command. For example:

```
Sudo /usr/openv/netbackup/bin/admincmd/bpimagerlist -l -d 04/17/2005 00:00:00
-e 04/23/2005 23:59:59 | grep IMAGE > imagelist_20050423.txt
```

In this example, the log file is `imagelist_20050423.txt`.

**Note** • You can name the file `imagelist.txt` or any other file name that includes a date in `yyyymmdd` format. That is, if you name the file anything other than `imagelist.txt`, the file name must include the date.

## NetBackup Log File Format

The NetBackup log file contains the following fields:

- IMAGE
- Client
- Date1
- Date2
- NBU version
- Backup ID
- Class
- Class type
- Proxy client
- Creator
- Schedule label
- Schedule type
- Retention level
- Backup time
- Elapsed time
- Expiration time
- Compressed
- Encrypted
- Kilobytes written
- Number of files
- Number of copies
- Number of fragments
- Files compressed
- File name
- Software version
- Name1
- bpimagerlist options
- Primary copy
- Image type
- True image recovery information
- True image recovery expiration
- Keywords
- MPX
- Extended security information
- Independent file restore from raw
- Image dump level
- File system only
- Previous block incremental time
- Block incremental full time
- Object description
- Request ID
- Backup status
- Backup copy

## Identifiers and Resources Collected from the NetBackup Log File

By default, the following field values in the log file are defined as chargeback identifiers and resource rate codes (see the `DefineIdentifier` and `DefineResource` methods in the `NetBackup.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

### Identifiers

- Feed (defined in the NetBackup collector job file)
- BackupClient
- BackupClass
- BackupType

### Resource Rate Codes

- NBKELPTM (Elapsed time in seconds)
- NBKKBWRI (Kilobytes written)
- NBKNMFIL (Number of files)

## Setting Up the NetBackup Collector

On the central CIMS Data Collectors server, set up an XML job file for the NetBackup collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector:

```

<Process id="NetBackup"
  description="Process for NetBackup Collector"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 NetBackup"
      type="ConvertToCSR"
      programName="NetBackup\NetBackup.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\LogFiles"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan NetBackup"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for NetBackup"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for NetBackup"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup NetBackup"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the NetBackup collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 7-3](#). These parameters are used by the conversion script, `NetBackup.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>If the log file name includes a date, the log date value is matched against the date in the file name to determine the file to be collected. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.</p> <p>If the log file is named <code>imagefile.txt</code>, this parameter is ignored.</p>
RetentionFlag	<p>This parameter is for future use.</p>
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="NetBackup"</code>, the output folder is <code>NetBackup</code>.</p>
LogFolder	<p>The location of the log file to be processed. The use of a UNC path for the log folder location is recommended.</p>

**Table 7-3 • NetBackup.wsf Parameters**

## Running the NetBackup Collector

Use CIMS Job Runner to run the NetBackup collector as described in [Running CIMS Job Runner](#) on page 2-93.

## Veritas

CIMS Lab provides a CIMS Data Collector for Veritas. For instructions on how to configure this collector, contact CIMS Lab.



---

# Network Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for network applications. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>NetFlow Data Collector</b> .....	<b>8-2</b>
Identifiers and Resources Collected from the NetFlow Data File .....	8-2
Setting Up the NetFlow Collector .....	8-3
Running the Netflow Collector .....	8-4
<b>Novell NetWare Data Collection</b> .....	<b>8-5</b>
<b>BindView Data Collector</b> .....	<b>8-5</b>

## NetFlow Data Collector

The CIMS Data Collector for NetFlow collects data that is contained in a data file produced by the CNS NetFlow Collection Engine application. This data file provides useful metrics such as packet, octet, and flow count.

---

**Note** • The data file name must include the date. Use the long form for the data file name (i.e., *export-resource-name\_yyyy\_mm\_dd.hhmm*) rather than the short format which does not include the date.

For more information about the data file, including naming and formatting conventions, go to [http://www.cisco.com/en/US/products/sw/netmgmtsw/ps1964/products\\_installation\\_and\\_configuration\\_guide\\_chapter09186a0080100259.html](http://www.cisco.com/en/US/products/sw/netmgmtsw/ps1964/products_installation_and_configuration_guide_chapter09186a0080100259.html).

---

## Identifiers and Resources Collected from the NetFlow Data File

By default, the following field values in the data file are defined as chargeback identifiers and resource rate codes (see the `DefineIdentifier` and `DefineResource` methods in the `Netflow.wsf` conversion script). The rate codes assigned to the resources *are not* pre-loaded in the `CIMSRate` table and must be added to the table as described in the *CIMS Server Administrator's Guide*.

### Identifiers

- Feed (defined in the NetFlow collector job file)
- srcaddr (source IP address)
- dstaddr (destination IP address)
- srcport (source port)
- dstport (destination port)
- prot (protocol byte)
- tos (type of service)

### Resource Rate Codes

- NFpkts (packet count)
- NFOctets (octet count)
- NFFlows (flow count)
- NFActTIM (active time)

## Setting Up the NetFlow Collector

On the central CIMS Data Collectors server, set up an XML job file for the NetFlow collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector:

```

<Process id="NetFlow"
  description="Process for NetFlow Collector"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 NetFlow"
      type="ConvertToCSR"
      programName="Netflow\Netflow.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\CallRecord"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan NetFlow"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for NetFlow"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for NetFlow"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
    <Step id="Cleanup"
      description="Cleanup NetFlow"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
      <Parameters>
        <Parameter DaysToRetainFiles="45"/>
      </Parameters>
    </Step>
  </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the NetFlow collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 8-1](#). These parameters are used by the conversion script, `Netflow.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	The log date specifies the date for the data file that you want to collect. For more information about using a log date, including valid log date values, see <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	The name of the server that contains the data file that you want to collect.  A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.  This parameter is included as an identifier in the CSR file.
OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.  The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="NetFlow"</code> , the output folder is <code>NetFlow</code> .
LogFolder	The location of the data file to be processed. The use of a UNC path for the data file location is recommended.

**Table 8-1 • Netflow.wsf Parameters**

## Running the Netflow Collector

Use CIMS Job Runner to run the NetFlow collector as described in [Running CIMS Job Runner](#) on page 2-93.

## Novell NetWare Data Collection

To collect usage data for Novell Netware, use the CIMS Windows Disk collector (see [Windows Disk Data Collector](#) on page 7-2). The CIMS Windows Disk collector scans a directory tree and provides a snapshot of the amount of disk space used by each top level folder within a specified drive or folder and the number of files within each folder (including all subfolders).

## BindView Data Collector

CIMS Lab provides a CIMS Data Collector for BindView. For instructions on how to configure this collector, contact CIMS Lab.

## ■ Network Data Collectors

---

*BindView Data Collector*

# Printer Data Collectors

This chapter contains instructions for setting up and running CIMS Data Collectors for printers. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>Windows Event Log Data Collector for Print</b> .....	<b>9-2</b>
Identifiers and Resources Collected by the CIMS Windows Event Log Collector .....	9-2
Setting Up the CIMS Windows Event Log Collector .....	9-3
Setting Event Log Security .....	9-8
Running the CIMS Windows Event Log Collector .....	9-8
Setting the Event Viewer Options for the System Event Log .....	9-9
<b>Windows Print Data Collector</b> .....	<b>9-10</b>
Creating a Log On User Account for the CIMS Windows Print Collector Service (Optional) ..	9-10
System Configuration Options for the CIMS Windows Print Collector .....	9-11
Installing the CIMS Windows Print Collector .....	9-14
Enabling Windows Print Logging .....	9-16
CIMS Windows Print Collector Log File Format .....	9-17
Identifiers and Resources Collected from the CIMS Windows Print Collector Log File .....	9-18
Setting Up the CIMS Windows Print Collector .....	9-20
Running the CIMS Windows Print Collector .....	9-25

## Windows Event Log Data Collector for Print

The CIMS Windows Event Log collector gathers printer events from the Windows System event log on a print server or servers. The collector provides useful metrics such as:

- The name of the user that ran the print job.
- The number of pages printed and the print job size in kilobytes.

The following sections provide instructions for setting up and running the CIMS Windows Event Log collector.

### Identifiers and Resources Collected by the CIMS Windows Event Log Collector

By default, the CIMS Windows Event Log collector creates the following chargeback identifiers and resource rate codes from the data collected. The rate codes are pre-loaded in the CIMSRate table.

Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>	
Feed (defined by the Feed parameter in the CIMSWinEventLog.xml file [see page 9-6])	—
UserName (the name of the user that ran the print job)	—
PrinterName (the name of the printer that produced the print job)	—
JobNumber (a job number assigned by the system)	—
JobName (an application-defined description of the document printed)	—
PortName (the printer port name)	—
<b>Resources</b>	
MS Windows Print Print KBytes	WPRTPRKB
MS Windows Print Page Count	WPRTPRPC

**Table 9-1 • Default CIMS Windows Event Log Identifiers and Resources**



## Setting Up the CIMS Windows Event Log Collector

On the central CIMS Data Collectors server, set up an XML job file for the CIMS Windows Event Log collector as described in *Creating Job Files* on page 2-25. The following is an example process for the CIMS Windows Event Log collector in the job file. This example collects print data from two servers: Server1 and Server2.

```
<Process id="CIMSWinEventLog"
description="Process for CIMS Windows Event Log Collector"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 CIMSWinEventLog"
type="ConvertToCSR"
programName="CIMSWinEventLog\CIMSWinEventLog.exe"
programType="console"
active="true">
      <GenerateExternalFile filename="%ProcessFolder%\CIMSWinEventLog.xml"
overwrite="true">
        <![CDATA[
          <?xml version="1.0" encoding="utf-8" ?>
          <CIMSCollectors version = "1.0">
            <Collectors>
              <Collector name="CIMSWinEventLog"
instanceName="Server1"
instanceDescription="Server1 Print Server"
Active="True">
                <Parameters>
                  <Parameter name="LogDate" value="%LogDate%"/>
                  <Parameter name="Retention" value="KEEP"/>
                  <Parameter name="Feed" value="Server1"/>
                  <Parameter name="OutputFolder"
value="%ProcessFolder%"/>
                  <Parameter name="LogSource"
value="Server1"/>
                  <Parameter name="LogType" value="server"/>
                  <Parameter name="EventType" value="Print"/>
                </Parameters>
              </Collector>
            </Collectors>
          </CIMSCollectors>
        ]]>
      </GenerateExternalFile>
    <Parameters>
      <Parameter UseStandardParameters="false"/>
      <Parameter XMLFileName="%ProcessFolder%\CIMSWinEventLog.xml"/>
      <Parameter CollectorName="CIMSWinEventLog"/>
    </Parameters>
  </Step>
```

```

<Step id="Server2 Collection"
description="Server2 CIMSWinEventLog"
type="ConvertToCSR"
programName="CIMSWinEventLog\CIMSWinEventLog.exe"
programType="console"
active="true">
<GenerateExternalFile filename="%ProcessFolder%\CIMSWinEventLog.xml"
overwrite="true">
  <![CDATA[
    <?xml version="1.0" encoding="utf-8" ?>
    <CIMSCollectors version = "1.0">
      <Collectors>
        <Collector name="CIMSWinEventLog"
instanceName="Server2"
instanceDescription="Server2 Print Server"
Active="True">
          <Parameters>
            <Parameter name="LogDate" value="%LogDate%"/>
            <Parameter name="Retention" value="KEEP"/>
            <Parameter name="Feed" value="Server2"/>
            <Parameter name="OutputFolder"
value="%ProcessFolder%"/>
            <Parameter name="LogSource"
value="Server2" />
            <Parameter name="LogType" value="server"/>
            <Parameter name="EventType" value="Print"/>
          </Parameters>
        </Collector>
      </Collectors>
    </CIMSCollectors>
  ]]>
</GenerateExternalFile>
<Parameters>
  <Parameter UseStandardParameters="false"/>
  <Parameter XMLFileName="%ProcessFolder%\CIMSWinEventLog.xml"/>
  <Parameter CollectorName="CIMSWinEventLog"/>
</Parameters>
</Step>
<Step id="Scan"
description="Scan CIMSWinEventLog"
type="Process"
programName="Scan"
programType="net"
active="true">
</Step>
<Step id="Process"
description="Standard Processing for CIMSWinEventLog"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
</Step>
<Step id="DatabaseLoad"
description="Database Load for CIMSWinEventLog"
type="Process"
programName="DBLoad"
programType="com"
active="true">
</Step>

```

```

        <Step id="Cleanup"
            description="Cleanup CIMSWinEventLog"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

Note that the ConvertToCSR steps contain the child elements GenerateExternalFile and Parameters.

When CIMS Job Runner is run, the GenerateExternalFile element dynamically creates an XML file (CIMSWinEventLog.xml) that contains parameters required by the CIMS Windows Event Log collector. The parameters include the LogDate and OutputFolder parameters, which must be included in the CIMSWinEventLog.xml file. For a description of the elements and attributes for this file, see *Defining the CIMSWinEventLog.xml File* on page 9-5.

---

**Note** • The file name CIMSWinEventLog.xml is defined for the ConvertToCSR step for both servers. The file defined in the first step will be overwritten by the file defined in the second step. If you do not want the final CIMSWinEventLog.xml file to appear in the process definition folder after the step has executed, use the GenerateExternalFile element attribute autoRemove (see page 2-60).

---

The Parameters element provides parameters for the step. For a description of these parameters, see page 2-87.

For a description of all other elements and attributes in the process, see *Creating Job Files* on page 2-25.

### Defining the CIMSWinEventLog.xml File

The CIMSWinEventLog.xml file contains the elements and attributes shown in Table 9-2. You can generate this file automatically by using the GenerateExternalFile element in the job file.

Element	Attributes	Description/Values
Collector	Collector Name	The collector name. <i>Do not change this parameter.</i>
	instanceName	The name of the instance for the collector. You can assign any name that has meaning for your organization. For example, the name of the server that you are collecting from.

Table 9-2 • CIMSWinEventLog.xml Elements and Attributes

Element	Attributes	Description/Values
Collector (continued)	instanceDescription	A description of the instance for the collector.
	Active	<p>Specifies whether the instance is included in processing. Valid values are:</p> <ul style="list-style-type: none"> <li>■ "true" (the instance is processed)</li> <li>■ "false" (the instance is not processed)</li> </ul> <p>The default is "true".</p>
Parameter	LogDate	<p>The log date specifies the date for the data that you want to collect.</p> <p>To use the log date passed from the command line, you need to use %LogDate% as the LogDate parameter in the CIMSEventLog.xml file (see the example on page 9-3).</p>
	Retention	This attribute is for future use.
	Feed	<p>The name of the server that contains the event log.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the OutputFolder parameter). This subfolder is used to store the initial CIMS Server Resource (CSR) file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>
	OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.

**Table 9-2 • CIMSWinEventLog.xml Elements and Attributes (Continued)**

Element	Attributes	Description/Values
Parameter (continued)	LogSource	<p>This attribute depends on the log type (see the LogType parameter).</p> <p>If you are collecting events from an archived event log, enter the path and file name of the archived file. Note that the file must be an event log file (.evt). You cannot use logs archived as .txt or .csv files. The use of a UNC path is recommended.</p> <p>If you are collecting events directly from the event log, enter the name of the server that contains the event log.</p> <p><b>Note:</b> If you are collecting data directly from the event log, see <a href="#">Setting Event Log Security</a> on page 9-8 for the required security permissions.</p>
	LogType	<p>The type of log. Valid values are:</p> <ul style="list-style-type: none"> <li>■ file (if collecting from archived .evt files)</li> <li>■ server (if collecting directly from the event log)</li> </ul>
	EventType	<p>Set this attribute to "Print". This value instructs the collector to gather data from events that are identified by <b>Print</b> in the <b>Source</b> column of the event log. This value is currently the only valid value.</p>

Table 9-2 • CIMSWinEventLog.xml Elements and Attributes (Continued)

### Using Smart Scan With the CIMS Windows Event Log Collector

The Smart Scan feature uses the Feed parameter values to determine the files to be scanned. For most collectors, the Feed parameter is provided in the ConvertToCSR step(s) in the job file. However, the Feed parameter for the CIMS Windows Event Log collector is provided in an external file. To use Smart Scan with this collector, you need to include either of the following in the ConvertToCSR step(s):

- A Feed parameter.
- Or
- A scanFile="file name" parameter where the file name includes the full path as shown in the preceding job file example. This enables Smart Scan to determine the location of the CSR file to be scanned.

For an example of a ConvertToCSR step with this configuration and for more information about Smart Scan, see [page 2-42](#).

## Setting Event Log Security

---

**Note** • This section is applicable only if you are collecting data directly from the event log.

---

To collect data from the event log, the Windows ID that you are using to run Windows Task Scheduler or the ID that you are using to log on to the computer running the CIMS Windows Event Log collector must have authority to read the event log.

### *If you are using Windows 2003 Server:*

Add the Windows user ID to the Performance Log Users group.

### *If you are using Windows Server 2000:*

Add the Windows user ID to one of the following groups:

- LocalSystem
- Administrator
- Server Operator (ServerOp)
- World

For more information, refer to the Microsoft Web site, [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/event\\_logging\\_security.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/event_logging_security.asp).

## Running the CIMS Windows Event Log Collector

Use CIMS Job Runner to run the CIMS Windows Event Log collector as described in *Running CIMS Job Runner* on page 2-93.

## Setting the Event Viewer Options for the System Event Log

Because the CIMS Windows Event Log collector gathers data from the Windows System event log on the print server, you should set the log size and overwrite options on the print server as described in the following steps. (Note that these steps are for the Microsoft Windows Server 2000 operating system. If you are using another operating system, refer to the Microsoft documentation if needed.)

- 1 In Windows Control Panel, double-click **Administrative Tools** ▶ **Event Viewer**.

The Event Viewer window appears.

- 2 Right-click **System**, and then click **Properties**.

The System Properties dialog box appears.

- 3 On the **General** tab, make sure that the **Maximum log size** is set to a size that will accommodate your collection schedule. For example, you might need to set a larger log size if you are collecting print events on a monthly schedule rather than a daily schedule.

- 4 Choose one of the following options under **When maximum log size is reached**:

- Click **Overwrite events older than** and enter a number 30 day longer than your collection schedule. For example, if you are collecting events daily, set the number to 31. If you are collecting events monthly, set the number to 60.
- Click **Do not overwrite events**. This option requires that you clear the log manually rather than automatically when the log is full.

To avoid deleting older events, do not click **Overwrite events as needed**.

- 5 Click **OK** when you are finished.

## Windows Print Data Collector

**Note** • In general, CIMS Lab recommends that you use the CIMS Windows Event Log collector (see [page 9-2](#)) rather than this collector. The Event Log collector gathers print data directly from the Windows System event log and does not require a separate installation on each server that you want to collect data from as the Print collector does. However, the Print collector provides more resources and identifiers than the Event Log collector. For assistance in determining which collector to use, contact CIMS Lab technical support.

---

The CIMS Windows Print collector gathers printer usage data for printers connected to a print server and produces a log file of the data (see [CIMS Windows Print Collector Log File Format](#) on page 9-17). This log file provides useful metrics such as:

- The name of the user that ran the print job.
- Number of pages submitted and printed and the print job size in kilobytes.
- Number of copies printed.

The following sections begin with important reference information for using the CIMS Windows Print collector, and then provide instructions for installing the collector, enabling print logging, and setting up and running the collector.

### Creating a Log On User Account for the CIMS Windows Print Collector Service (Optional)

The CIMS Windows Print collector includes a Windows *service* that supports the collector. By default, the service runs under the Local System user account. CIMS Lab recommends that you use this default account; however, you can run the service using a user or group account that has been granted the following security policies:

- Debug programs
- Log on as a service

You can assign these policies to a local account or a domain account. If you use a local account, you need to set the policies at both the domain and local level if you are using a domain. Policies for the domain override local policies.

The steps to assign the required domain and local security policies are the same as those outlined for the CIMS Windows Process collector on [page 3-2](#).



## System Configuration Options for the CIMS Windows Print Collector

You can use any of the following system configurations to process the log files produced by the CIMS Windows Print collector. These configurations are presented in order of recommendation. The first configuration is the most simple and secure.

### Configuration 1: Pulling Log Files to the Central Server

In this configuration, the log files are written to a log folder on the server running the CIMS Windows Print collector and then pulled to the central CIMS Data Collectors server for processing.

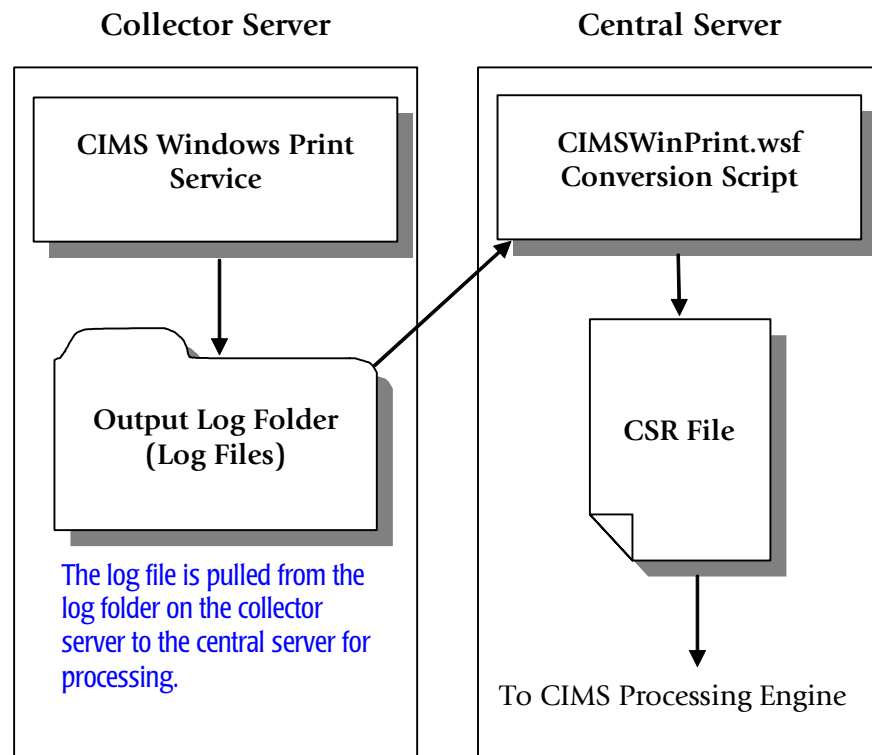


Figure 9-1 • System Configuration 1

For an example of the job file XML that supports this configuration, see [Job File Example for Configurations 1 and 2](#) on page 9-20.

### Configuration 2: Writing Log Files Directly to the Central Server

In this configuration, the log files are written directly to a log folder on the central CIMS Data Collectors server for processing.

**Note** • A disadvantage of this configuration is that if the network connection between the collector server and the central server is down, the log files are lost.

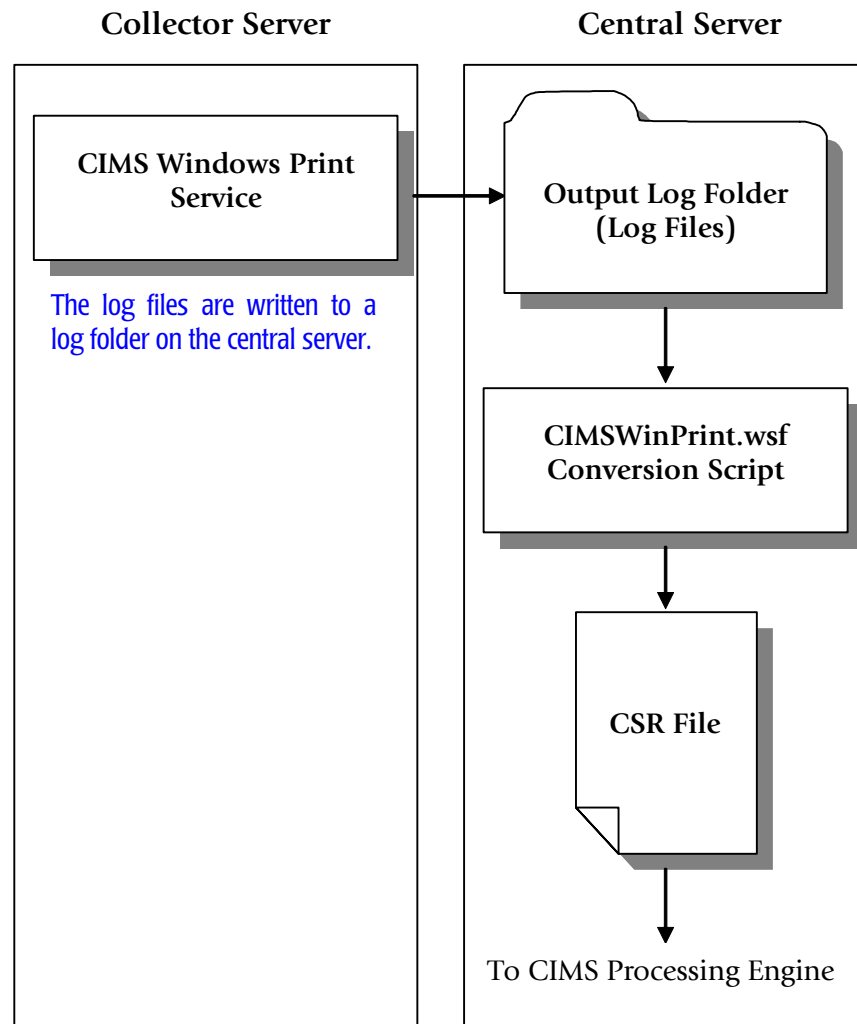


Figure 9-2 • System Configuration 2

For an example of the job file XML that supports this configuration, see [Job File Example for Configurations 1 and 2](#) on page 9-20.

### Configuration 3: Generating CSR Files on the Collector Server

**Note** • This configuration is usually not recommended. For more information, contact CIMS Lab.

In this configuration, the log files are written to a log folder on the server running the CIMS Windows Print collector. The CIMSWinPrint.wsf script is also run on this server. The output CSR records can be written on the server running the collector *or* on the central CIMS Data Collectors server; however, the CSR files must be processed by CIMS Processing Engine on the central server.

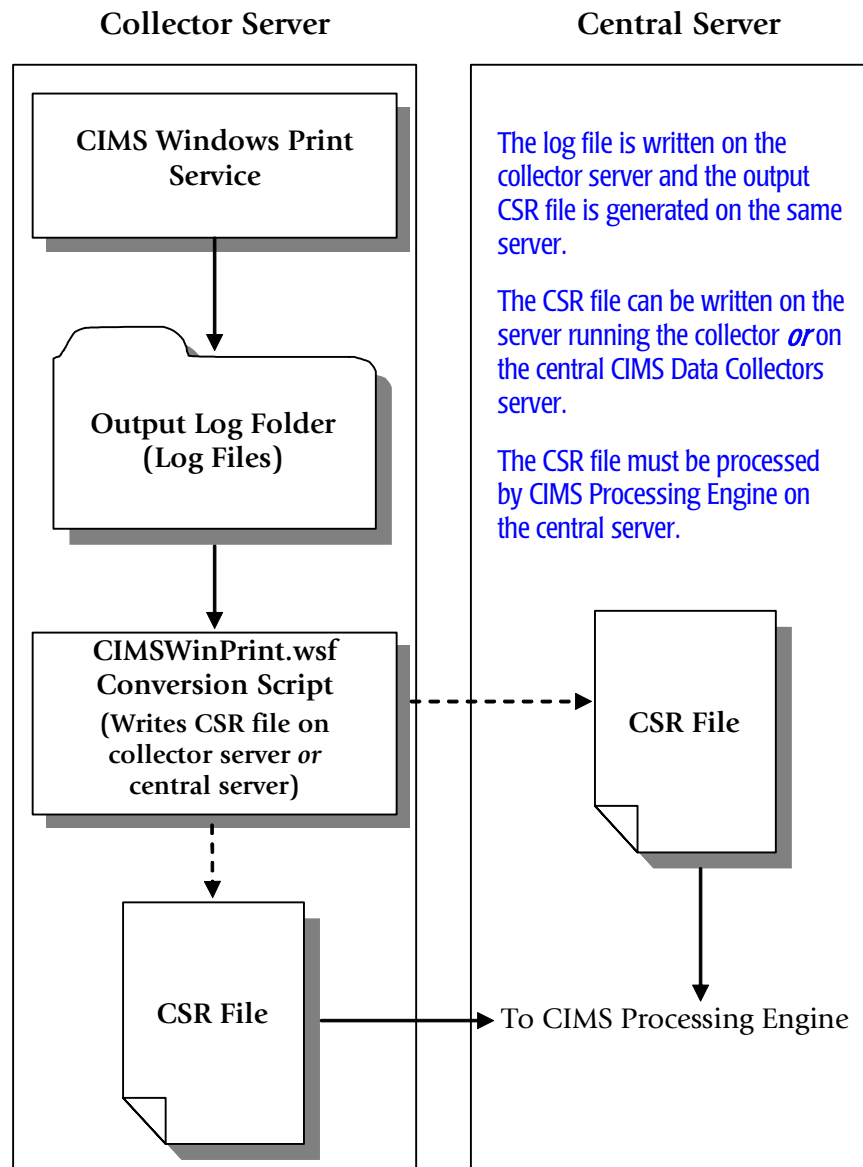


Figure 9-3 • System Configuration 3

For an example of the job file XML that supports this configuration, see [Configuration 3: Generating CSR Files on the Collector Server](#) on page 9-13.

## Installing the CIMS Windows Print Collector

To use the CIMS Windows Print collector, you must have the CIMS Windows Print collector installed on the central CIMS Data Collectors server as described on [page 2-2](#).

In addition to installation on the central server, you need to install the CIMS Windows Print collector on each print server that you want to collect data from.

CIMS Lab provides a simple setup program, CIMSWinPrintSetup.exe, in the Collectors\CIMSWinPrint folder on the central server. You can copy this program to a central location such as a network drive so that you can quickly install the CIMS Windows Print collector on other computers.

### **To install the CIMS Windows Print collector:**

---

**Note** • These following steps are also applicable if you are upgrading to a new version or release of the CIMS Windows Print collector.

---

- 1 Log on to Windows as an Administrator.
- 2 Click the Windows **Start** button, and then click **Run**.
- 3 Enter the path to the setup program CIMSWinPrintSetup.exe and then click **OK**. If you installed the CIMS Windows Print collector on the central CIMS Data Collectors server, you can find this program in the Collectors\CIMSWinPrint folder.

The setup wizard appears.

- 4 Click **Next**.
- 5 Choose the default location for installation (C:\Program Files\CIMSLab) or click **Change** to choose another location. After making your selection, click **Next**.
- 6 In most cases, you will accept the default install features (i.e., leave the **Print Collector** check box selected and all other check boxes unselected).

Although it is not common, you might want to generate CSR files on the server running CIMS Windows Print collector as shown in [Configuration 3: Generating CSR Files on the Collector Server](#) on page 9-13. In this case, you also need to select the **Aggregation Engine** check box to install CIMS Aggregation Engine (CIMSAggregation.dll).

- 7 Click **Next**, and then click **Install**.
- 8 Click **Finish** to complete the installation.

## Components Installed by CIMSWinPrintSetup.exe

The CIMSWinPrintSetup.exe setup program installs the following components:

- **The CIMS Windows Print Collector service.** This is a Windows service that supports the collector. To view Windows services, in Windows Control Panel, open **Administrative Tools ▶ Services**.
- **The CIMS Windows Print collector.** This installs the following components in the Collectors\CIMSWinPrint folder created during installation:
  - The executable program for the collector, CIMSWinPrintService.exe.
  - An executable program, CIMSWinPrintServiceLog.exe, that is used by CIMS Lab for troubleshooting purposes. For more information about this program, contact CIMS Lab.
  - The executable program for the collector's administrative program, CIMSWinPrintServiceAdmin.exe.
  - The conversion script, CIMSWinPrint.wsf. In most cases, this file is used on the central CIMS Data Collectors server and is not needed on other computers. The exception is if you are converting log files to CSR files on the computer running the CIMS Windows Print collector (see [page 9-13](#)).

If you selected the **Aggregation Engine** check box in the setup wizard, CIMS Aggregation Engine is also installed.

The installation does not include CIMS Processing Engine, which processes the CSR files created by CIMS Aggregation Engine and loads the output data into the database. To process CSR files, you need to process the files on the central CIMS Data Collectors server.

## Enabling Windows Print Logging

The CIMS Windows Print collector tracks print jobs for selected printers connected to the print server and enters the usage data for each job as a record in the log file.

The CIMS Windows Print collector includes an easy-to-use GUI program for configuring and enabling the collection process. To use this program, click the **Start** menu, and then click **Programs** ▶ **CIMS Server** ▶ **Collectors** ▶ **CIMS Windows Print Administrator** and set the following options:

- **Log file path.** Enter the path to the folder that you want to store the print log files in. If the file does not exist, you will be asked if you want to create the path. Click **Yes**.

The log file folder can be on the computer running the CIMS Windows Print collector or on the central CIMS Data Collectors server, depending on the system configuration that you are using for collection and processing (see *System Configuration Options for the CIMS Windows Print Collector* on page 9-11). You should create this folder in a location where you keep data that is backed up.

---

**Important!** • Do not set the log file path to the `Processes\CIMSWinPrint\<feed>` folder. The feed folder should contain only CSR files.

---

The default path is `C:\Program Files\CIMSLab\CIMSWinPrintLogs` (if you installed CIMS Data Collectors in the default location). The use of a UNC path for the log file location is recommended.

- **Log file prefix.** The default name for the log file is `CIMSPrintLog-yyyymmdd.txt`. You can use the default prefix `CIMSPrintLog-` or replace it with the prefix of your choice (or no prefix).
- **Use Local Time in output records.** If this check box is selected (the default), the local time set for the computer is used in the date and time fields in the log file. If this check box is cleared, Universal Time Coordinate (*UTC*) time is used in the log file.

---

**Note** • The date in the log file name always reflects local time, regardless of whether Use Local Time is selected.

---

- **Monitored Printer List.** The printers that you want to monitor for data collection. Click **Add** or **Remove** to add or delete printers from this list.

When you click **Add**, the Select Printers dialog box appears. After the collector searches for shared local and network printers, you can do one of the following:

- Enter a printer name in the **Printer** box.
- Select one of the printers listed in **Available Printers**.

- **Retry unavailable printers every.** If a printer in the **Monitored Printer List** is unavailable, enter the amount of time in seconds, minutes, or hours that you want the CIMS Windows Print collector to recheck for availability.
- **Control Service.** Click this button to open the Service Control dialog box to start or stop the CIMS Windows Print collector. You can also start and stop the collector from Windows Control Panel and then click the **Refresh** button in the Service Control dialog box to make the change in the collector.

## CIMS Windows Print Collector Log File Format

The following table describes the record fields in the log file produced by the CIMS Windows Print collector.

Field Name	Description/Values
RecordType	The record type is J for job.
JobID	The job ID assigned by the system.
MachineName	The name of the computer that generated the print job.
UserName	The name of the user that ran the print job.
PrinterName	The name of the printer that produced the print job.
PrinterServerName	The name of the print server for the printer that produced the print job.
PrinterShareName	The share name of the printer that produced the print job.
JobName	An application-defined description of the document printed.
PortName	The printer port name.
SubmitKBytes	The number of kilobytes submitted.
PrintKBytes	The number of kilobytes printed.
SubmitPageCount	The number of pages submitted.
PrintPageCount	The number of pages printed.
Copies	The number of copies printed.
Priority	The priority of the print job in the print server queue.
SubmitDateTime	The date and time that the print job was submitted.
CompleteDateTime	The date and time that the print job was completed.

**Table 9-3 • CIMS Windows Print Collector Log File Format**

Field Name	Description/Values
DuplexType	Indicates whether the print job is single- or double-sided.
FormName	The form name. If there is no form name a hyphen (-) appears.
Orientation	Portrait or landscape.
PrintQuality	The print quality in dots per inch (DPI).
PaperSource	The source of the paper, for example, automatically select, manual feed, or tray number.
PaperSize	The paper size.
ColorOutput	Specifies whether the print output was in color.

**Table 9-3 • CIMS Windows Print Collector Log File Format (Continued)**

## Identifiers and Resources Collected from the CIMS Windows Print Collector Log File

By default, the following fields in the CIMS Windows Print collector log file are defined as the chargeback identifiers and resources (see the `DefineIdentifier` and `DefineResource` methods in the `CIMSWinPrint.wsf` conversion script). The rate codes assigned to the resources are pre-loaded in the `CIMSRate` table.

Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
<b>Identifiers</b>		
—	Feed (passed from the <code>JobCIMSWinPrint</code> script)	—
MachineName	MachineName	—
UserName	User	—
PrinterName	PrinterName	—
PrinterServerName	Server	—
PrinterShareName	PrinterShareName	—
JobName	JobName	—
PortName	PortName	—
Priority	Priority	—

**Table 9-4 • Default CIMS Windows Print Identifiers and Resources**



Log File Field	Identifier Name or Resource Description in CIMS Server	Assigned Rate Code in CIMS Server
DuplexType	DuplexType	—
FormName	FormName	—
Orientation	Orientation	—
PrintQuality	PrintQuality	—
PaperSource	PaperSource	—
ColorOutput	ColorOutput	—
<b>Resources</b>		
SubmitKBytes	MS Windows Print Submit KBytes	WPRTSBKB
PrintKBytes	MS Windows Print Print KBytes	WPRTPRKB
SubmitPageCount	MS Windows Print Submit Page Count	WPRTSBPC
PrintPageCount	MS Windows Print Page Count	WPRTPRPC
Copies	MS Windows Print Copies	WPRTCOPY

**Table 9-4 • Default CIMS Windows Print Identifiers and Resources (Continued)**

## Setting Up the CIMS Windows Print Collector

You will need to set up an XML job file on the central CIMS Data Collectors server for all system configurations. If you are using the system configuration shown in [Configuration 3: Generating CSR Files on the Collector Server](#) on page 9-13, you also need to set up a job file on the server running the CIMS Windows Print collector.

The following sections provide job file examples by system configuration type.

### Job File Example for Configurations 1 and 2

On the central CIMS Data Collectors server, set up an XML job file for the CIMS Windows Print collector as described in [Creating Job Files](#) on page 2-25. The following is an example process for the CIMS Windows Print collector in the job file. Note that the location of the log folder is defined by the `LogFolder` parameter. Depending on the system configuration that you are using, the value for this parameter will be the path to the log folder on the server running the collector (Configuration 1) or the path to the log folder on the central server (Configuration 2).

```
<Process id="CIMSWinPrint"
description="Process for CIMS Windows Print Collection"
active="true">
  <Steps>
    <Step id="Server1 Collection"
description="Server1 CIMSWinPrint"
type="ConvertToCSR"
programName="CIMSWinPrint\CIMSWinPrint.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\CIMSWinPrintLogs"/>
      </Parameters>
    </Step>
    <Step id="Scan"
description="Scan CIMSWinPrint"
type="Process"
programName="Scan"
programType="net"
active="true">
    </Step>
    <Step id="Process"
description="Standard Processing for CIMSWinPrint"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
    <Step id="DatabaseLoad"
description="Database Load for CIMSWinPrint"
type="Process"
programName="DBLoad"
programType="com"
active="true">
    </Step>
  </Steps>
</Process>
```

```

        <Step id="Cleanup"
            description="Cleanup CIMSWinPrint"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the CIMS Windows Print collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 9-5](#). These parameters are used by the conversion script, `CIMSWinPrint.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

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 <a href="#">Specifying Log Dates for Collection</a> on page 2-3.
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the log file that you want to process. If the log file is on the same server as the <code>CIMSWinPrint.wsf</code> script used to convert the file, you can also use "Self" and the server name is defined automatically (see the example on <a href="#">page 9-23</a>).</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <a href="#">Feed Subfolder</a> on page 2-13). This is the CSR file that is processed by the Scan program.</p> <p>This parameter is included as an identifier in the CSR file.</p>

**Table 9-5 • CIMSWinProcess.wsf Parameters**

Parameter	Description/Values
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>By default, the output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id="CIMSWinPrint"</code>, the output folder is <code>CIMSWinPrint</code>.</p> <p>This parameter is required only if you are running the <code>CIMSWinPrint.wsf</code> script on one server, but want to send CSR files to a process definition folder on another server. (This configuration is not common.) In this case, you need to provide the path to the process definition folder.</p>
LogFolder	<p>The location of the log file to be processed.</p>

---

**Table 9-5 • CIMSWinProcess.wsf Parameters (Continued)**

### Job File Examples for Configuration 3

The job file XML for this system configuration differs depending on whether CSR files are written to the server running the CIMS Windows Print collector or the central CIMS Data Collectors server.

#### To write the CSR files to the server running the CIMS Windows Process collector:

On the computer running the CIMS Windows Print collector, set up a job file as described in [Creating Job Files](#) on page 2-25. The process for the collector in the job file should contain only a `ConvertToCSR` step and a `FileTransfer` step as shown in the following example.

In this example, the log files are written to the collector server (Server1). CSR files created from the log files will be copied from the `CIMSWinPrint\Server1` folder on the collector server to the `CIMSWinPrint\Server1` folder on the central server (CIMS).

The `%LogDate_End%` variable in the `FileTransfer` from parameter specifies that CSR files that contain a date matching the last day of the `LogDate` parameter are copied. For example, if the `LogDate` parameter is the default `PREDAY`, CSR files with the previous day's date are copied. (For more information about the `LogDate` parameter, see [Specifying Log Dates for Collection](#) on page 2-3.)

```
<Process id="CIMSWinPrint"
  description="Process for CIMS Windows Print Collection"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 CIMSWinPrint"
      type="ConvertToCSR"
      programName="CIMSWinPrint\CIMSWinPrint.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Self"/>
        <Parameter LogFolder="\\Server1\CIMSWinPrintLogs"/>
      </Parameters>
    </Step>
    <Step id="FileTransfer"
      description="Transfer CSR Files"
      type="Process"
      programName="FileTransfer"
      programType="net"
      active="true">
      <Parameters>
        <Parameter type="Windows"/>
        <Parameter from="\\Server1\CIMSWinPrint\Server1\
          %LogDate_End%.txt"
          to="\\CIMS\CIMSWinPrint\Server1"
          action="Copy"
          overwrite="true"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

On the central CIMS Data Collectors server, set up a job file that does not contain the ConvertToCSR step for the collector (i.e., Scan is the first step). For example:

```
<Process id="CIMSWinPrint"
  description="Process for CIMS Windows Print Collection"
  active="true">
  <Steps>
    <Step id="Scan"
      description="Scan CIMSWinPrint"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for CIMSWinPrint"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    :
    :
```

***To write the CSR files to the central CIMS Data Collectors server:***

On the computer running the CIMS Windows Print collector, set up a job file as described in *Creating Job Files* on page 2-25. The process for the collector in the job file should contain only a ConvertToCSR step as shown in the following example.

In this example, the log files are written to the collector server (Server1). The path for the log file folder is specified by the LogFolder parameter.

The CSR files created from the log files are written to the central CIMS Data Collectors server (CIMS) using the path specified by the OutputFolder parameter and the value of the Feed parameter. That is, CSR files will be written to the feed subfolder Server1 in the CIMSWinProcess process definition folder on the central server.

```
<Process id="CIMSWinPrint"
  description="Process for CIMS Windows Print Collection"
  active="true">
  <Steps>
    <Step id="Server1 Collection"
      description="Server1 CIMSWinPrint"
      type="ConvertToCSR"
      programName="CIMSWinProcess\CIMSWinPrint.wsf"
      programType="wsf"
      active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
        <Parameter LogFolder="\\Server1\CIMSWinPrintLogs"/>
        <Parameter OutputFolder="\\CIMS\CIMSWinPrint"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

On the central CIMS Data Collectors server, set up a job file that does not contain the ConvertToCSR step for the collector (i.e., Scan is the first step). For example:

```
<Process id="CIMSWinProcess"
  description="Process for CIMS Windows Print Collection"
  active="true">
  <Steps>
    <Step id="Scan"
      description="Scan CIMSWinPrint"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for CIMSWinPrint"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    :
    :
```

## Running the CIMS Windows Print Collector

Use CIMS Job Runner to run the CIMS Windows Print collector as described in *Running CIMS Job Runner* on page 2-93.

## ■ **Printer Data Collectors**

---

*Windows Print Data Collector*



---

# Transactions Collector

This chapter contains instructions for setting up and running the CIMS Data Collector for transactions. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>About Transactions</b> .....	10-2
<b>About the CIMSTransaction Table</b> .....	10-2
Identifiers and Resources Collected from the CIMSTransaction Table .....	10-4
<b>Setting Up the Transactions Collector</b> .....	10-4
<b>Running the Transactions Collector</b> .....	10-6

## About Transactions

In some circumstances, you might want to generate a CIMS Server Resource (CSR) file for occurrences that are not contained in a usage metering file. For example, you might want to generate a CSR file to apply a credit for an overcharge or to charge for a one time occurrence such as the cost of providing a computer to a new employee.

In these cases, you can create a miscellaneous, recurring, or credit transaction in CIMS Server Web Reporting. These transactions contain the chargeback information that you want to include in a CSR file. For more information about transactions, refer to the *CIMS Server Web Reporting User's Guide*.

Transactions are stored in the CIMSTransaction table in the CIMS Server database. The Transactions collector is used to collect, convert, and process the transactions on a monthly basis.

The following sections provide instructions for setting up and running the Transactions collector.

## About the CIMSTransaction Table

The CIMSTransaction table contains the following fields.

Field Name	Field Description
TransactionUID	The unique identifier for the transaction.
AccountCode	The account code for the transaction.
TransactionType	The transaction type: <ul style="list-style-type: none"><li>■ M (Miscellaneous)</li><li>■ R (Recurring)</li><li>■ C (Credit)</li></ul>
ShiftCode	The shift code for the transaction.
RateCode	The rate code for the transaction.
ResourceAmount	The amount of the transaction.

---

**Table 10-1 • CIMSTransaction Table Fields**

Field Name	Field Description
Frequency1	<p>Applicable only to recurring transactions. The frequency that the transaction should occur (every month, every 6 months, etc.). Frequency is based on the calendar year (January–December)</p> <ul style="list-style-type: none"> <li>■ 1 (monthly)</li> <li>■ 2 (every other month)</li> <li>■ 3 (every quarter)</li> <li>■ 4 (every four months)</li> <li>■ 6 (every six months)</li> <li>■ 12 (once a year)</li> </ul>
Frequency2	<p>Applicable only to recurring transactions. The period in which the transaction should be processed. This value corresponds to the value in the Frequency1 field. For example, if the value in the Frequency1 field is 6, a value of 1 in this field indicates the first month of a 6 month period (January or July).</p>
FromDate/ToDate	<p>Applicable only to miscellaneous and credit transactions. The date range that the transaction is to occur.</p>
DateTimeSent	<p>The date and time that the transaction was exported to a flat file.</p>
DateTimeModified	<p>The date and time that the transaction was last modified.</p>
DateTimeEntered	<p>The date and time that the transaction was created.</p>
DateTimeStartProcessing	<p>Applicable only to recurring transactions. The first day that the transaction will be processed.</p>
DateTimeStopProcessing	<p>Applicable only to recurring transactions. The last day that the transaction will be processed.</p>
UserID	<p>The CIMS Server Web Reporting user ID of the person who entered the transaction.</p>
Note	<p>A description of the transaction.</p>
DateTimeDeleted	<p>The date and time that the transaction was deleted.</p>

**Table 10-1 • CIMSTransaction Table Fields (Continued)**

## Identifiers and Resources Collected from the CIMSTransaction Table

By default, the Transactions collector creates the following chargeback identifiers from the transactions collected. The collector uses the rate code that is contained in the transaction records. This rate code is already loaded in the CIMSRate table.

CIMSTransaction Table Field	Identifier Name
AccountCode	Account_Code
Note	Description
DateTimeSent	DateTimeSent
DateTimeModified	DateTimeModified
DateTimeEntered	DateTimeEntered
UserID	UserID

**Table 10-2 • Default Transaction Identifiers and Resources**

## Setting Up the Transactions Collector

On the central CIMS Data Collectors server, set up an XML job file for the Transactions collector as described in *Creating Job Files* on page 2-25. The following is an example process for the collector in the job file.

```

<Process id="Credits"
description="Credit Transactions"
joblogShowStepOutput="true"
joblogShowStepParameters="true"
active="true">
  <Steps>
    <Step id="Credit Collection"
description="Credit Collection"
type="ConvertToCSR"
programName="Transactions\Transactions.wsf"
programType="wsf"
active="true">
      <Parameters>
        <Parameter Feed="Credit"/>
        <Parameter LogDate="CURMON"/>
      </Parameters>
    </Step>

    <Step id="Process"
description="Standard Processing for Credit Collection"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
      </Step>

    <Step id="DatabaseLoad"
description="Database Load for Credit Collection"
type="Process"
  
```

```

        programName="DBLoad"
        programType="com"
        active="true">
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Transactions collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 10-3](#) on page 10-5. These parameters are used by the conversion script, `Transactions.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The month that the transaction is scheduled to occur. Valid values are:</p> <ul style="list-style-type: none"> <li>■ premon (previous month)</li> <li>■ curmon (current month)</li> <li>■ yyyypp (year and period [1–13])</li> </ul> <p>The <code>LogDate</code> parameter cannot be passed from the command prompt—it must be included in the job file for the Transactions collector (see the example on <a href="#">page 10-4</a>).</p> <p><b>Note:</b> The period for a credit or miscellaneous transaction is the period that the transaction occurred as defined in the <code>CIMSCalendar</code> table. The period for a recurring transaction is the period that is set for the transaction.</p>
RetentionFlag	This parameter is for future use.
Feed	You can enter any value for this parameter. Although a subfolder of the same name is automatically created in the process definition folder (see the <code>OutputFolder</code> parameter), it is not used. CSR files are placed directly within the process definition folder.

**Table 10-3 • Transactions.wsf Parameters**

Parameter	Description/Values
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <code>Process id</code> attribute in the job file. For example, if the <code>Process id= "Transactions"</code>, the output folder is <code>Transactions</code>.</p>
DataSourceID (optional)	<p>The CIMS Data Source ID for the CIMS Server database that contains the transactions.</p> <p>If this parameter is not provided, the CIMS Data Source that is set as the <code>Web/collector</code> default in the CIMS Data Source Maintenance dialog box in CIMS Server Administrator is used.</p> <p>To use a CIMS Data Source other than the default, set this parameter to the appropriate CIMS Data Source ID.</p>

---

**Table 10-3 • Transactions.wsf Parameters (Continued)**

## Running the Transactions Collector

Use CIMS Job Runner to run the Transactions collector as described in [Running CIMS Job Runner](#) on page 2-93.

---

# Mainframe Data Collectors

This chapter contains instructions for setting up and running collection of the output files produced by CIMS Mainframe Data Collector and Chargeback System. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>About Mainframe Data Collection .....</b>	<b>11-2</b>
<b>Setting Up Collection for CSR or CSR+ Files .....</b>	<b>11-3</b>
<b>Setting Up Collection for CIMS Ident, Detail, and Summary Files .....</b>	<b>11-5</b>
<b>Running the Mainframe Collection Process .....</b>	<b>11-6</b>

## About Mainframe Data Collection

CIMS Mainframe Data Collector and Chargeback System produces the following files that can be loaded into the CIMS Server database:

- **CIMS Server Resource (CSR) files.** These files are produced by CIMS Mainframe 11.6 and earlier. These are the standard CSR files used by CIMS Server. CSR files require processing by CIMSAct and CIMSBill in CIMS Server before they can be loaded into the CIMS Server database.

Or

- **CIMS Server Resource Plus (CSR+) files.** 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.

CSR+ files require processing by CIMSAct and CIMSBill in CIMS Server before they can be loaded into the CIMS Server database.

Or

- **CIMS Ident, Detail, and Summary files.** These files are produced by CIMS Mainframe 12.0 and later. These files do not require processing by CIMSAct and CIMSBill. They are loaded directly into the CIMS Server database.

These files are sent via FTP or another method from CIMS Mainframe to the appropriate process definition folder on the CIMS Server system (see [Process Definitions \(Processes Folder\)](#) on page 2-12). CIMS Lab provides default mainframe process definition folders named Mainframe, MainframeCICS, MainframeDB2, etc., that may be used to store these files.

CSR and CSR+ files must be sent to a feed subfolder within the process definition folder (see [Feed Subfolder](#) on page 2-13). For example, you might have a subfolder named AL90 in the Mainframe folder to store CSR+ files for the AL90 system. The CSR and CSR+ files are named *yyyymmdd.txt*.

CIMS Ident, Detail, and Summary files must be sent directly to the process definition folder. The CIMS Ident, Detail, and Summary files are named *Ident\_yyyyymmdd.txt*, *BillDetail\_yyyyymmdd.txt*, and *BillSummary\_yyyyymmdd.txt*.

For more information about creating and transferring mainframe CSR, CSR+, or CIMS Ident, Detail, and Summary files, refer to the *CIMS Mainframe Data Collector and Chargeback System User Guide*.

The following sections describe how to set up the collection process for CSR+ files and CIMS Ident, Detail, and Summary files.



## Setting Up Collection for CSR or CSR+ Files

On the central CIMS Data Collectors server, set up an XML job file as described on [Creating Job Files](#) on page 2-25. The job file must contain a `Process` element for each process definition folder that you want to collect CSR or CSR+ files from as shown in the following example.

Note that the first step in the Mainframe process is the `WaitForFile` step. This optional step instructs CIMS Job Runner to wait for a CSR or CSR+ file that matches the `fileName` attribute before continuing processing. A UNC path is used in the `fileName` attribute to define the CSR/CSR+ path and name.

This example processes a CSR+ file. CIMS Job Runner will wait for a CSR+ file that matches the `%LogDate_End%` variable value to be sent to the feed subfolder AL90. (For more information about `%LogDate_End%` and other variables, see [page 2-61](#)).

The `ConvertToCSR` step is not required because the CSR+ file is already in the required format .

```
<Process id="Mainframe"
  description="Process for Mainframe Collection"
  active="true">
  <Steps>
    <Step id="WaitForFile"
      description="Wait for CSR+ File"
      type="Process"
      programName="WaitFile"
      programType="net"
      active="true">
      <Parameters>
        <Parameter pollingInterval="60"/>
        <Parameter fileName="\\Server1\AL90\%LogDate_End%.txt"/>
        <Parameter timeOutDateTime="%RNDATE% 13:50:59"/>
      </Parameters>
    </Step>
    <Step id="Scan"
      description="Scan Mainframe"
      type="Process"
      programName="Scan"
      programType="net"
      active="true">
    </Step>
    <Step id="Process"
      description="Standard Processing for Mainframe"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for Mainframe"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
    </Step>
  </Steps>
```

```
        <Step id="Cleanup"
            description="Cleanup Mainframe"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>
<Process id="MainframeCICS"
    description="Process for Mainframe CICS Collection"
    active="true">
    <Steps>
        <Step id="Scan"
            description="Scan MainframeCICS"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
        </Step>
        <Step id="Process"
            description="Standard Processing for MainframeCICS"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
        </Step>
        <Step id="DatabaseLoad"
            description="Database Load for MainframeCICS"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup MainframeCICS"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>
```

## Setting Up Collection for CIMS Ident, Detail, and Summary Files

On the central CIMS Data Collectors server, set up an XML job file as described on [Creating Job Files](#) on page 2-25. The job file must contain a `Process` element for each process definition folder that you want to collect CIMS Ident, Detail, and Summary files from as shown in the following example.

Note that the first step in the `MainframeDB2` process is the `WaitForFile` step. This optional step instructs CIMS Job Runner to wait for the CIMS Ident, Detail, and Summary files that match the `fileName` attribute before continuing processing. Paths for these files are not required because the files must be in the process definition folder defined by the `Process id` attribute. For more information about `%LogDate_End%` and other variables that you can use in the file names, see [page 2-61](#).

The `ConvertToCSR`, `Scan`, and `Process` steps are not required because the files do not require conversion or processing and are ready to be loaded into the CIMS Server database. CIMS Mainframe has performed all processes such as account code conversion and processing through `CIMSAct` and `CIMSBill`.

```
<Process id="MainframeDB2"
  description="Process for Mainframe DB2 Collection"
  active="true">
  <Steps>
    <Step id="WaitForFile"
      description="Wait for Summary, Detail, and Ident Files"
      type="Process"
      programName="WaitFile"
      programType="net"
      active="true">
      <Parameters>
        <Parameter pollingInterval="60"/>
        <Parameter fileName="BillSummary_%LogDate_End%.txt"/>
        <Parameter fileName="BillDetail_%LogDate_End%.txt"/>
        <Parameter fileName="Ident_%LogDate_End%.txt"/>
        <Parameter timeOutDateTime="%RNDATE% 13:50:59"/>
      </Parameters>
    </Step>
    <Step id="DatabaseLoad"
      description="Database Load for MainframeDB2"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
      <Parameters>
        <Parameter useDatedFiles="true"/>
      </Parameters>
    </Step>
```

```
<Step id="Cleanup"
      description="Cleanup MainframeDB2"
      type="Process"
      programName="Cleanup"
      programType="net"
      active="true">
  <Parameters>
    <Parameter DaysToRetainFiles="45"/>
  </Parameters>
</Step>
</Steps>
</Process>
```

## Running the Mainframe Collection Process

To run the collection process for files produced by CIMS Mainframe Data Collector and Chargeback System, see [Running CIMS Job Runner](#) on page 2-93.

---

## UNIX Data Collectors

This chapter contains instructions for setting up and running collection of the CIMS Server Resource (CSR) files produced by CIMS Data Collector for UNIX. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on page 2-3 before continuing with the collector-specific information in this chapter.

<b>About UNIX Data Collection .....</b>	<b>12-2</b>
<b>Setting Up Collection for CSR Files .....</b>	<b>12-2</b>
<b>Running the UNIX Collection Process .....</b>	<b>12-4</b>

## About UNIX Data Collection

CIMS Data Collector for UNIX produces CSR files that can be processed by CIMS Server or the CIMS Mainframe Data Collector and Chargeback System application. This chapter discusses how to collect and process the CSR files for input into the CIMS Server database. For the steps required to process a CSR file using the CIMS Mainframe system, refer to the *CIMS Mainframe Data Collector and Chargeback System User Guide*.

The CSR files are sent via FTP or Secure Shell from the CIMS UNIX system to the appropriate process definition folder on the CIMS Server system (see *Process Definitions (Processes Folder)* on page 2-12). CIMS Lab provides default UNIX process definition folders named UnixOS, UnixDB2, UnixORA, etc., that may be used to store these files.

CSR files must be sent to a feed subfolder within the process definition folder (see *Feed Subfolder* on page 2-13). The feed subfolder might represent the server from which the CSR files are sent. For example, if the server UnixServer is sending CSR files for a Unix Oracle database, you might create the feed subfolder UnixServer in the UnixORA process definition folder. The CSR files are named *yyyymmdd.txt*.

For more information creating and transferring UNIX CSR files, refer to the *CIMS Data Collector for UNIX Installation and Getting Started Guide*.

The following section describes how to set up the collection process for the CSR files.

## Setting Up Collection for CSR Files

On the central CIMS Data Collectors server, set up an XML job file as described on *Creating Job Files* on page 2-25. The job file must contain a Process element for each process definition folder that you want to collect CSR files from as shown in the following example.

Note that the ConvertToCSR step is not required because the CSR file is already in the required format.

```
<Process      id="UnixORA"
             description="Process for Unix Oracle Collection"
             active="true">
  <Steps>
    <Step      id="Scan UnixORA"
             description="Scan UnixORA"
             type="Process"
             programName="Scan"
             programType="net"
             active="true">
  </Step>
    <Step      id="Process"
             description="Standard Processing for UnixORA"
             type="Process"
             programName="SingleProcessStep"
             programType="com"
             active="true">
  </Step>
```

```

        <Step id="DatabaseLoad"
            description="Database Load for UnixORA"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup Unix Oracle"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>
<Process id="UnixFS"
    description="Process for Unix Filesystem Collection"
    active="true">
    <Steps>
        <Step id="Scan"
            description="Scan UnixFS"
            type="Process"
            programName="Scan"
            programType="net"
            active="true">
        </Step>
        <Step id="Process"
            description="Standard Processing for UnixFS"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
        </Step>
        <Step id="DatabaseLoad"
            description="Database Load for UnixFS"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup UnixFS"
            type="Process"
            programName="Cleanup"
            programType="net"
            active="true">
            <Parameters>
                <Parameter DaysToRetainFiles="45"/>
            </Parameters>
        </Step>
    </Steps>
</Process>

```

## **Running the UNIX Collection Process**

To run the collection process for CSR files produced by CIMS Data Collector for UNIX, see *Running CIMS Job Runner* on page 2-93.



## Other Data Collectors

### SAP, Shiva, and Evolve Data Collectors

CIMS Lab provides a CIMS Data Collector for SAP, Shiva, and Evolve. For instructions on how to configure these collectors, contact CIMS Lab.

## ■ Other Data Collectors

---

*SAP, Shiva, and Evolve Data Collectors*

## CIMS Universal Data Collector

This chapter describes how to use the CIMS Universal Data Collector. You should have a good understanding of the CIMS Data Collector system architecture as described in the *CIMS Data Collectors Architecture* section beginning on [page 2-3](#) before continuing with the collector-specific information in this chapter.

<b>About CIMS Universal Data Collector</b> .....	14-2
<b>The Data Conversion Process</b> .....	14-2
<b>Creating a Conversion Definition Using CIMS Conversion Builder</b> .....	14-3
Creating a Definition file .....	14-3
Opening a Conversion Definition .....	14-24
Saving a Conversion Definition .....	14-24
Viewing Conversion Definitions .....	14-24
Running CIMS Conversion Engine .....	14-24
<b>Setting Up and Running the Universal Collector</b> .....	14-25
Adding Resource Rate Codes to the CIMSRate Table .....	14-25
Setting Up the Universal Collector .....	14-25
Running the Universal Collector .....	14-27
<b>Example Files</b> .....	14-28
Log File—SodaLog.txt .....	14-28
Conversion Definition File—SodaLogDef.txt .....	14-29
Output File—CurrentCSR.txt .....	14-37

## About CIMS Universal Data Collector

In addition to the data collectors described in the preceding chapters, CIMS Lab provides a universal data collector, CIMS Universal Data Collector, for applications that do not have a specific CIMS Data Collector. The Universal collector uses the CIMS Conversion Engine utility to convert *any usage metering data from any application* to a CIMS Server Resource (CSR) file.

## The Data Conversion Process

The following are the files and CIMS components used in the data conversion process.

### Application Usage Metering File

The usage metering file can be either of the following:

- Any ASCII file with either fixed length fields or delimited fields (for example, a log file). Each file entry must be on a single line.
- Any database log file.

### Conversion Definition

The conversion definition is a file that defines the format of the usage metering file as well as the data that will appear in the output CSR file. You can create conversion definitions using the CIMS Conversion Builder application (see [Creating a Definition file](#) on page 14-3) or you can create definition files using a text editor such as Notepad (see [Conversion Definition Viewed in Notepad](#) on page 14-36).

If you have multiple usage metering files with different formats, you need to create a separate conversion definition for each file type.

### Universal Collector

The Universal collector calls the CIMS Conversion Engine (ProcConvEng.wsf) to convert the usage metering file. The steps required to set up and run the Universal collector are provided in [Setting Up and Running the Universal Collector](#) on page 14-25.

### CIMS Conversion Engine

CIMS Conversion Engine converts the data in the usage metering file based on the conversion definition. The output from CIMS Conversion Engine is a CSR file.

## Creating a Conversion Definition Using CIMS Conversion Builder

You can create a conversion definition as a text file (see *Conversion Definition Viewed in Notepad* on page 14-36); however, the CIMS Conversion Builder GUI provides a much simpler way to create a conversion definition. CIMS Conversion Builder also provides data validation features that ensure the conversion definition can be processed successfully by CIMS Conversion Engine. This section describes how to create a conversion definition using CIMS Conversion Builder.

To start CIMS Conversion Builder, click **Start ▶ Programs ▶ CIMS Server ▶ Conversion Builder** (if you installed CIMS Data Collectors in the default location). Or, from the CIMS Server Administrator main window, click **Chargeback Administration ▶ Processing ▶ Conversion Builder**.

### Creating a Definition file

CIMS Conversion Builder provides the following tabs that walk you through each of the required and optional steps for creating a conversion definition:

- **Input tab.** Defines the input usage metering file data.
- **Output tab.** Defines the output file.
- **Fields tab.** Defines the fields in the usage metering file.
- **Identifiers tab.** Defines the usage metering file fields to be used as identifiers in the output file.
- **Resources tab.** Defines the usage metering file fields to be used as resources in the output file.
- **Date/Time tab.** Defines the start and end date and time that appear in the output file.
- **Shifts tab.** Defines whether shift processing is enabled.

The following sections provide descriptions for each of the options on these tabs. For each tab option, the corresponding conversion definition statement is also provided.

---

**Note** • CIMS Lab provides a sample conversion definition (SodaLogDef.txt) in Processes\Universal where the folder Processes can be in any location (see *About the Processes Folder* on page 2-12). See page 14-29 for examples of these tabs as they appear for SodaLogDef.txt.

---

## Input Tab

Use the **Input** tab to enter the parameters for the usage metering file as shown in the following table. All parameters are required unless noted otherwise.

For an example of a configured **Input** tab, see [page 14-29](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Description	Description=<description of this definition file>	Briefly describes the purpose of the conversion definition. The field size is approximately 100 bytes.  A description is optional and has no impact on the conversion process.
Input Type	ProcessType=	The type of data to be processed: data in an ASCII text file or data extracted from a database query.
<ul style="list-style-type: none"> <li>■ Delimited ASCII Text File (default)</li> <li>■ Fixed-length ASCII Text File</li> <li>■ ODBC Query</li> <li>■ Microsoft Access Database Query</li> </ul>	<ul style="list-style-type: none"> <li>■ DELIMITED</li> <li>■ FIXED</li> <li>■ ODBCQUERY</li> <li>■ MSACCESS</li> </ul>	<p>For ASCII files, delimited means the usage metering file record has fields separated by a delimiter, such as a Comma Separated Values (CSV) file. Fixed means the file record has fixed-length fields.</p> <p>The input type determines which of the following processing options appear. If the input type is an ASCII text file, continue to <a href="#">ASCII Text File</a> on page 14-5. If the input type is a database query, skip to <a href="#">Database Query</a> on page 14-7.</p>

**Table 14-1 • Input Tab**

CIMS Conversion Builder Option	Definition File Statement	Description
<b>ASCII Text File</b>		
Input Filename	InputFile=<path and file name of input file>	This is the path and name of the input usage metering file. The maximum path that can be specified is approximately 250 bytes.  <b>Important:</b> If you are running CIMS Conversion Engine from the Universal collector, the input file name must be defined in the collector's job file rather than in the conversion definition (see <i>Creating a Job File</i> on page 14-25). The file name entered in the job file will override the value in the conversion definition.
Record Delimiter ■ NEWLINE (default) ■ BLANKLINE ■ FORMFEED	RecDelimiter = ■ NEWLINE ■ BLANKLINE ■ FORMFEED	The character used to delimit records (normally NEWLINE).  If fields are terminated by a new line, then set the record delimiter to BLANKLINE.  You can select a delimiter from the list, or you can enter the ASCII keyboard character(s) for other delimiters. For example, ^I for a tab.
Field Delimiter (does not appear for Fixed-length input type) ■ COMMA (default) ■ TAB ■ SEMICOLON ■ COLON ■ NEWLINE ■ SPACE	Delimiter= ■ COMMA ■ TAB ■ SEMICOLON ■ COLON ■ NEWLINE ■ SPACE ■ <any character literal>	The character used to delimit fields in a usage metering file.  A field delimiter is required only for delimited files.  You can select a delimiter from the list, or you can enter the ASCII keyboard character(s) for other delimiters. For example, a forward slash (/).

Table 14-1 • Input Tab (Continued)

CIMS Conversion Builder Option	Definition File Statement	Description
<p>Text Field Qualifier (does not appear for Fixed-length input type)</p> <ul style="list-style-type: none"> <li>■ DOUBLEQUOTE (default)</li> <li>■ QUOTE</li> <li>■ NONE</li> </ul>	<p>TextQualifier=</p> <ul style="list-style-type: none"> <li>■ DOUBLEQUOTE</li> <li>■ QUOTE</li> <li>■ NONE</li> <li>■ &lt;any character literal&gt;</li> </ul>	<p>The character used for fields with embedded delimiter characters.</p> <p>For example, if the field delimiter is COMMA and the field value is "1,345" then the text field qualifier is DOUBLEQUOTE. The quotation marks in this case mark the beginning and ending of the field value.</p> <p>Quote indicates a single quote qualifier.</p> <p>A text field qualifier is required only for delimited files.</p> <p>You can select a qualifier from the list, or you can enter the ASCII keyboard character(s) for other delimiters. For example, an asterisk (*).</p>
<p>Skip Initial Lines</p>	<p>InitialSkipLineCnt =n</p>	<p>The number of lines to skip before beginning to process a usage metering file. You can select a number from the drop-down list or type a number.</p> <p>This is useful in situations where there are a number of header lines preceding the actual data. The default is 0 (skip no lines).</p>

**Table 14-1 • Input Tab (Continued)**



CIMS Conversion Builder Option	Definition File Statement	Description
<b>Database Query</b>		
ODBC Data Source (appears when ODBC Query input type is selected)	InputFile=<name of ODBC data source>	<p>The ODBC Data Source for the database to be queried. The data source must be listed in the Windows ODBC Data Source Administrator.</p> <p>Do one of the following:</p> <ul style="list-style-type: none"> <li>■ Type the data source name (the data source must be listed in the Windows ODBC Data Source Administrator).</li> <li>■ Click <b>Browse</b> to select the data source from the Select ODBC Data Source dialog box. If the data source that you want to use is listed in the dialog box, click the database, and then click <b>OK</b>.</li> <li>■ If the data source is not listed in the dialog box, click the <b>ODBC Data Source Administrator</b> button, click the <b>System DSN</b> tab, and click <b>Add</b> to add the data source.</li> </ul>
Access Database (appears when Microsoft Access Database Query input type is selected)	InputFile=<path and file name of database>	The access database to be queried. The maximum path that can be specified is approximately 250 bytes.
ODBC User ID	OdbcUid=<user ID>	The user ID for the database (if required).
ODBC User Password	OdbcPwd=<encrypted user password>	The user password for the database (if required). The password is encrypted.
ODBC SQL Query	OdbcQuery=<SQL query>	The database query.

**Table 14-1 • Input Tab (Continued)**

## Output Tab

Use the **Output** tab to enter the parameters for the output file as shown in the following table. All parameters are required unless noted otherwise.

For an example of a configured **Output** tab, see [page 14-30](#)

CIMS Conversion Builder Option	Definition File Statement	Description
Output Filename	OutputFile=<path and file name of generated output file>	<p>This is the path and name of the output file. The output file must be stored in a feed subfolder within the process definition folder (see <i>Feed Subfolder</i> on page 2-13).</p> <p>The maximum path that can be specified is approximately 250 bytes.</p> <p><b>Important:</b> If you are running CIMS Conversion Engine from the Universal collector, the output file name is automatically defined as <code>yyyymmdd.txt</code>. This is the initial CSR file that is processed by the Scan program. This file path and name will override the value in the conversion definition.</p>
Output Record Type ■ CIMS Server Resource Record (default) ■ CIMS Transaction Record	OutRecType= ■ CBS ■ TRANS	<p>The CIMS record type in the output file.</p> <p>If you are running CIMS Conversion Engine from the Universal collector, you cannot create transaction records. However, you can create transaction records if you run CIMS Conversion Engine from CIMS Conversion Builder (see <a href="#">page 14-24</a>).</p> <p><b>Note:</b> Although CIMS Transaction records are supported, CIMS Lab recommends that you use the CIMS Server Resource (CSR) records. CSR records are more flexible and efficient than transaction records.</p>

Table 14-2 • Output Tab

CIMS Conversion Builder Option	Definition File Statement	Description
Resource Header (appears when CIMS Server Resource Record is selected)	UnivHdr=<field name>	<p>The resource header defines the source of data. A resource header is not available in all usage metering files and is not required.</p> <p>Depending on whether the usage metering file contains a header, you can do the following:</p> <ul style="list-style-type: none"> <li>■ If the records within the file <i>do not</i> contain a header, you can add a header here if you want a header to appear in the output file. Otherwise, leave this box blank.</li> <li>■ If the records within the file <i>do</i> contain a header, you can select the header field from the drop down list (if the field is entered in the <b>Fields</b> tab), type the field name, or leave the field blank (if you do not want the header to appear in the output file).</li> </ul>

Table 14-2 • Output Tab (Continued)

CIMS Conversion Builder Option	Definition File Statement	Description
Output standard server identifiers (appears when CIMS Server Resource Record is selected)	WriteStandardServerIdentifiers= <input type="checkbox"/> YES <input type="checkbox"/> NO	<p>When this check box is selected, the <b>identifiers</b> <code>SourceName</code> and <code>SourceLine</code> are added to the output file.</p> <p><code>SourceName</code> shows the path for the source (for fixed-length, comma delimited or Access query input types) or the name of the data source (for the ODBC query type).</p> <p><code>SourceLine</code> shows the line of the usage metering file that produced the record.</p> <p>Standard identifiers are optional. The values for these identifiers can be lengthy. If the length of the output records is a consideration, leave this check box clear (the default).</p>
Audit Code Default (appears when CIMS Transaction Record is selected)	AuditCodeDefault= <string literal>	<p>A string that is used to hold a default audit code value (see <a href="#">page 14-19</a>). A default audit code is optional.</p> <p>The default audit code can be a maximum of eight characters and simply serves as a user-defined field that helps to identify the record (i.e., an employee code, service code, etc.). The audit code does not affect data processing in any way.</p>

**Table 14-2 • Output Tab (Continued)**

## Fields Tab

Use the **Fields** tab to define the fields in the usage metering file as shown in the following table.

The required parameters depend on the input type.

- For fixed-length usage metering files, the **Field Name**, **Starting Column** (starting position for the field), and **Length** parameters are required. The **Type** parameter is also required for date and time fields if you are using date and time fields as the start and/or date time in the output file records (see *Date/Time Tab* on page 14-21).
- For all other usage metering files, only the **Field Name** is required with the exception that the **Type** is also required for date and time fields if you are using the date and time fields as the start and/or date time in the output file records.

---

**Note** • If the input type is a database query, click **Populate Field List Using Query**. CIMS Conversion Builder automatically populates the fields.

---

For an example of a configured **Fields** tab, see [page 14-31](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Field Number	Fieldn	An incrementing sequence number used to uniquely identify a field.
Field Name	<field name>	The field name. The name must be a single word or abbreviation (for example, ACCTCD for account code).
Starting Column	COL(n)	The starting position for the field.  The starting position number is required for fixed-length files. It is optional for delimited files.
Length	LEN(n)	The length of the field. The length is required for fixed-length files. It is optional for delimited files.
Implied Decimals	DEC(n)	The number of decimal digits for the field. For example, if the field value in the usage metering file is 10000 and the implied decimal count is 2, the resulting value in the output file is 100.00.

**Table 14-3 • Fields Tab**

CIMS Conversion Builder Option	Definition File Statement	Description
Type (Date/Time)	TYPE(date or time)	<p>Time and date fields require a TYPE declaration specifying the format of the time and date as they appear in the usage metering file.</p> <p>The type format used for time and date fields is dependent on whether the time and date are fixed length or variable length.</p> <p><b>Time Fields</b></p> <ul style="list-style-type: none"> <li>■ <b>Fixed length.</b> A fixed length time format is one in which there are a fixed number of digits for the time. For example, 12:34, 01:15, etc.</li> </ul> <p>Fixed length time fields in a usage metering file <i>do not</i> require a separator character. However, if the field includes a separator character, for example, 12:34, you need to include the character in the type format. If the field does not include a separator, for example, 1234, the separator character is optional.</p> <p>You can use hour (H), minutes (M), and seconds (S) in the following format: HH, MM, SS (seconds are optional). The format must be preceded by "T-".</p> <p>Examples: T-HHMMSS, T-HH:MM</p>

Table 14-3 • Fields Tab (Continued)

CIMS Conversion Builder Option	Definition File Statement	Description
Type (Date/Time) (continued)		<p><b>Time Fields (continued)</b></p> <ul style="list-style-type: none"> <li>■ <b>Variable length.</b> A variable length time format is one in which there <i>is not</i> a fixed number of digits for the time. For example, 12:34, 1:15 (no preceding 0), etc.</li> </ul> <p>Variable length time fields in a usage metering file require a separator character and the character must be included in the type format.</p> <p>You can use hour (H), minutes (M), and seconds (S) in the following format: H, M, and S (seconds are optional). The format must be preceded by "T-".</p> <p>Examples: T-H:M, T-H:M:S</p> <p><b>Date Fields</b></p> <ul style="list-style-type: none"> <li>■ <b>Fixed length.</b> A fixed length date format is one in which there are a fixed number of digits for the date. For example, 12252005, 01012006, etc.</li> </ul> <p>Fixed length date fields in a usage metering file <i>do not</i> require a separator character. However, if the field includes a separator character, for example, 12/25/2005, you need to include the character in the type format. If the field does not include a separator, for example, 12252005, the separator character is optional.</p> <p>You can use any combination of year (Y), month (M), and day (D) in the following format: YY or YYYY, MM or MMM, and DD. The format must be preceded by a "D-".</p> <p>Examples: D-YYYYMMDD, D-MM/DD/YYYY</p>

Table 14-3 • Fields Tab (Continued)

CIMS Conversion Builder Option	Definition File Statement	Description
Type (Date/Time) (continued)		<p><b>Date Fields (continued)</b></p> <ul style="list-style-type: none"> <li>■ <b>Variable length.</b> A variable length date format is one in which there <i>is not</i> a fixed number of digits for the date. For example, 12/25/2005, 1/01/2006 (no preceding 0), etc.</li> </ul> <p>Variable length date fields in a usage metering file require a separator character and the character must be included in the type format.</p> <p>You can use any combination of year (Y), month (M), and day (D) in the following format: Y, M, and D. The format must be preceded by a "D-".</p> <p>Examples: D-Y/M/D, D-M/D/Y</p> <p>If a year contains only two digits, the century is determined by the following:</p> <ul style="list-style-type: none"> <li>■ Years 0–29 are assumed to occur in the 2000s (2000–2029)</li> <li>■ Years 30–99 are assumed to occur in the 1900s (1930–1999)</li> </ul>

**Table 14-3 • Fields Tab (Continued)**



CIMS Conversion Builder Option	Definition File Statement	Description
Filter	FILTERPATTERN(reg expression)	<p>A regular expression or a literal that the field must match, otherwise the record is rejected.</p> <p>Regular expressions are used frequently in some utilities and programming languages such as grep, sed, awk, and Perl.</p> <p>The regular expression FILTERPATTERN is a subset of full regular expressions available in other tools and can consist of the following metacharacters:</p> <ul style="list-style-type: none"> <li>^-Matches to beginning of field</li> <li>\$-Matches to end of field</li> <li>*-Matches zero or more occurrences of the preceding literal</li> <li>.-Matches any character</li> <li>!-If this is the first character in an expression, it negates the outcome of the regular expression. That is, the expression is not matched.</li> </ul> <p><b>Example</b></p> <p>A usage metering file contains records with one of two account codes: 01100 or 01200. If you want just those records that contain the account code 01200, you could use the regular expressions ^012, 200\$, 01*200, 0.2, !01100 (among others) or the literal 01200.</p>

Table 14-3 • Fields Tab (Continued)

CIMS Conversion Builder Option	Definition File Statement	Description
Parse—Character	PARSECHAR (character)	<p>The character used to split a string in the field. For example, to split a URL, enter the / character for this parameter. The character delimits the end of a word.</p> <p><b>Parse—Character and Parse—Word Number</b> work together to parse a “word” from a string.</p> <p>For example, if you want only <code>cimsnt.asp</code> in the following example:</p> <pre>http://www.cimslab.com/ cimsnt.asp</pre> <p>The parse character is / and the parse word is the fourth word in the string as follows:</p> <p>Word 1=http:  Word 2=null  Word 3=www.cimslab.com  Word 4=cimsnt.asp</p> <p>In this case, you enter a 4 in the <b>Parse—Word Number</b> box.</p>
Parse—Word Number	PARSEWORD(n)	<p>The number of the word in the string that should be split by the parse character (see the preceding example for <b>Parse—Character</b>) and returned as the field value. The character delimits the end of a word.</p> <p>If the value for the parse word number is greater than the number of words indicated by the parse character, the last word in the string is returned. For example, if you entered a parse word number of 5 for the preceding example, the field value would be <code>cimsnt.asp</code> (there is no fifth word).</p>

**Table 14-3 • Fields Tab (Continued)**

CIMS Conversion Builder Option	Definition File Statement	Description
Insert Field		Inserts a field above the selected field.
Remove Field		Removes the selected field.
Populate Field List Using Query (appears when a database query input type is selected)		Automatically populates the field list with fields from the database. You can then change the field names if needed.

**Table 14-3 • Fields Tab (Continued)**

### Identifiers Tab

Use the **Identifiers** tab to define the fields that are identifiers. Identifier fields are used as literals or lookup keys in the account code conversion in CIMS Server.

For an example of a configured **Identifiers** tab, see [page 14-32](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Field Number	IDFIELDn	An incrementing sequence number used to uniquely identify an identifier.
Field Name	<field name>	The field used as an identifier.  To select the fields that you want to use as identifiers, click the drop-down arrow in the <b>Field Name</b> box. (The drop-down arrow appears when you click the box.)
Insert Field		Inserts a field above the selected field.
Remove Field		Removes the selected field.

**Table 14-4 • Identifiers Tab**

## Resources Tab

Use the **Resources** tab to define the fields that represent resource usage. For example, a field that represents CPU time, transactions processed, or lines printed.

For an example of a configured **Resources** tab, see [page 14-33](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Field Number	RSFIELDn	An incrementing sequence number used to uniquely identify a resource.
Field Name	<field name>	<p>The field containing a resource.</p> <p>To select the fields that you want to use as resources, click the drop-down arrow in the <b>Field Name</b> box. (The drop-down arrow appears when you click the box.)</p>
Rate Code	RATECODE(code)	<p>The rate code represents the resource units being reported by the field.</p> <p>To enter a rate code for the field, click the <b>Rate Code</b> box.</p> <p>Click the (...) button and do one of the following:</p> <ul style="list-style-type: none"> <li>■ If a field within the usage metering file contains the rate code, click the field name.</li> <li>■ If the rate code is contained in the CIMSRate table of the CIMS Server database (see <a href="#">ODBC Data Source Name</a> on page 14-20), click the existing rate code.</li> <li>■ If the rate code is not contained in the usage metering file or the database, type the rate code name in the lower box. Do not use the same name for both the resource field and the rate code.</li> </ul> <p><b>Important:</b> If you select a rate code from the usage metering file or create a new rate code, <i>you must add the rate code to the CIMSRate table</i>. Rate codes that do not appear in the CIMSRate table are not included in CIMS Server invoices and other reports.</p>

Table 14-5 • Resources Tab

CIMS Conversion Builder Option	Definition File Statement	Description
Audit Code (appears when CIMS Transaction Record is selected as the output record type)	AUDITCODE(code)	<p>A literal value specifying the audit code used to track this resource value. An audit code is optional.</p> <p>The default audit code can be a maximum of eight characters and simply serves as a user-defined field that helps to identify the record (i.e., an employee code, service code, etc.). The audit code does not affect data processing in any way.</p> <p>To enter a audit code for the field, click the <b>Audit Code</b> box.</p> <p>Click the (...) button and do one of the following:</p> <ul style="list-style-type: none"> <li>■ If a field within the usage metering file contains the audit code, click the field name.</li> <li>■ If you want to use the default audit code entered in the <b>Output</b> tab, click DEFAULT.</li> <li>■ If you want to enter an audit code, type the code in the lower box.</li> </ul>
Insert Field		Inserts a field above the selected field.
Remove Field		Removes the selected field.

**Table 14-5 • Resources Tab (Continued)**

<b>CIMS Conversion Builder Option</b>	<b>Definition File Statement</b>	<b>Description</b>
ODBC Data Source Name	RateOdbcDsn=<name of database>	<p>The default ODBC data source for the database that contains the CIMSRate table is CIMSServer. If you want to use the CIMSRate table from another database, do one of the following:</p> <ul style="list-style-type: none"> <li>■ Type the data source name (the data source must be listed in the Windows ODBC Data Source Administrator).</li> <li>■ Click <b>Browse</b> to select the data source from the Select ODBC Data Source dialog box. If the data source that you want to use is listed in the dialog box, click the database, and then click <b>OK</b>.</li> <li>■ If the data source is not listed in the dialog box, click the <b>ODBC Data Source Administrator</b> button, click the <b>System DSN</b> tab, and click <b>Add</b> to add the data source.</li> </ul>
ODBC User ID	OdbcUid=<user ID>	The user ID for the database (if required).
ODBC Password	OdbcPwd=<encrypted user password>	The user password for the database (if required). The password is encrypted.

**Table 14-5 • Resources Tab (Continued)**

### Date/Time Tab

Use the **Date/Time** tab to define the start and end date and time that appear in the output file records.

For an example of a configured **Date/Time** tab, see [page 14-34](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Record Date Low and High	RecDateLo= ■ SYSTEM ■ <field name> ■ RNDATE ■ CURDAY ■ CURWEK ■ CURMON ■ PREDAY ■ PREWEK ■ PREMON RecDateHi= ■ SYSTEM ■ <field name>	Determines the start and end date that appear in the output file records. You can select one of the following: <ul style="list-style-type: none"> <li>■ System Date. The computer system date is used. This is the default.</li> <li>■ One of the following date keywords. If you select a keyword in the <b>Record Date Low</b> box, you cannot select values in the <b>Record Date High</b> box or the <b>Record Time</b> boxes. These boxes are unavailable.                             <ul style="list-style-type: none"> <li>• <b>Run Date (Today)</b>. The start and end date is the current day.</li> <li>• <b>Previous Day to Current Day</b>. The start date is the previous day and the end date is the current day.</li> <li>• <b>Current Week/Month</b>. The start date is the first day of the current week/month and the end date is the last day of the current week/month.</li> <li>• <b>Previous Day</b>. The start and end date are the previous day.</li> <li>• <b>Previous Week/Month</b>. The start date is the first day of the previous week/month and the end date is the last day of the previous week/month.</li> </ul> </li> <li>■ A date field (if defined in the <b>Fields</b> tab). The value in the date field is used as the date.</li> </ul>

Table 14-6 • Date/Time Tab

CIMS Conversion Builder Option	Definition File Statement	Description
Record Time Low and High	RecTimeLo= ■ SYSTEM ■ <field name> ■ ENTIRE RecTimeHi= ■ SYSTEM ■ <field name> ■ ENTIRE	Determines the start and end time that appear in the output file records. Note that if a keyword is selected in the <b>Record Date Low</b> box, the <b>Record Time</b> boxes are unavailable.  You can select one of the following: <ul style="list-style-type: none"> <li>■ <b>System Time.</b> The computer system time is used. This is the default.</li> <li>■ <b>A time field</b> (if defined in the <b>Fields</b> tab). The value in the time field is used as the time.</li> <li>■ <b>Entire Day.</b> Defines the start time as 00:00:00 and the end time as 23:59:59.</li> </ul>

**Table 14-6 • Date/Time Tab (Continued)**



### Shifts Tab

Use the **Shift** tab to define whether shift processing is enabled. In shift processing, a shift character is entered in the Shift Code field (for CSR records) or appended to the existing rate code (for CIMS Transaction records). Using shifts enables you to charge different rates for different work shifts.

When entering shifts:

- You may enter a maximum of 5 shifts per day.
- The shift characters can be a numeric value 1–9, and the times must be listed in 4-character, 24-hour format.

For an example of a configured **Shifts** tab, see [page 14-35](#).

CIMS Conversion Builder Option	Definition File Statement	Description
Shift Processing Enabled	ShiftsEnabled= <ul style="list-style-type: none"> <li>■ YES</li> <li>■ NO</li> </ul>	If the check box is selected, the use of shifts is enabled. If the check box is clear, the use of shifts is not enabled.
Shift Field	ShiftField=<field>	The name of a time field which is used to generate a shift character in the output file. If a time field is not specified, the output file record start time is used to generate the shift character (see <a href="#">Date/Time Tab</a> on page 14-21).
Shift Char	Shift<day>= DEFINE <shift char> <end time> [ <shift char> <end time> ...]	The number (1–9) that represents the shift, for example, 1 for the first shift, 2 for the second shift, etc.
End Time		The time that the shift ends.

**Table 14-7 • Shifts Tab**

## Opening a Conversion Definition

To open a conversion definition, click **File ▶ Open Conversion Definition**.

## Saving a Conversion Definition

To save a new conversion definition, click **File ▶ Save As**. To save changes to an existing definition, click **File ▶ Save**.

## Viewing Conversion Definitions

You can view the conversion definition, usage metering file, and output file for the current definition directly from CIMS Conversion Builder. To view a file, click **File ▶ View File Type**.

## Running CIMS Conversion Engine

Once you have created a conversion definition for a usage metering file, you can run CIMS Conversion Engine directly from CIMS Conversion Builder to ensure that the output file contains the data that you want.

To run CIMS Conversion Engine from CIMS Conversion Builder, click **File ▶ Run Conversion**. The output file is created and placed in the location specified on the **Output** tab (see *Output Tab* on page 14-8).

## Setting Up and Running the Universal Collector

This section provides the information you need to set up and run the Universal collector.

### Adding Resource Rate Codes to the CIMSRate Table

Because the resources collected by CIMS Data Collector are user-defined in the conversion definition and not pre-defined by CIMS Lab, you need to add the rate codes for the resources to the CIMSRate table.

Rate codes that do not appear in the CIMSRate table are not included in CIMS invoices and other reports. You cannot load an output file into the CIMS Server database until at least one rate code from the file is added to the CIMSRate table. To add rate codes, refer to the *CIMS Server Administrator's Guide*.

### Setting Up the Universal Collector

#### Creating a Process Definition Folder

Each conversion definition must reside in a separate process definition folder. For example, if you are collecting usage metering files from a Unisys system or an Informix database, you might have a process definition folders named `Unisys` and `Informix`. Unlike other CIMS Data Collectors in which CIMS Job Runner will create the process definition folder from the Process ID in the job file, you need to create a process definition folder containing the conversion definition before you run CIMS Job Runner.

#### Creating a Job File

On the central CIMS Data Collectors server, set up an XML job file for the Universal collector as described in see [Creating Job Files](#) on page 2-25. The following is an example process for the Universal collector in the job file. This example processes the sample log file `SodaLog.txt` using the conversion definition `SodaLogDef.txt` (see [Example Files](#) on page 14-28).

```
<Process      id="Universal"
             description="Process for SodaLog"
             active="true">
  <Steps>
    <Step      id="Server1 Collection"
             description="Server1 SodaLog"
             type="ConvertToCSR"
             programName="Universal\Universal.wsf"
             programType="wsf"
             active="true">
      <Parameters>
        <Parameter Feed="SodaLog"/>
        <Parameter ConvEngDefName="%ProcessFolder%\SodaLogDef.txt"/>
        <Parameter InputFileName="%ProcessFolder%\SodaLog.txt"/>
        <Parameter OutputFolder="%ProcessFolder%"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

```

    </Step>
    <Step id="Scan"
        description="Scan SodaLog"
        type="Process"
        programName="Scan"
        programType="net"
        active="true">
    </Step>

    <Step id="Process"
        description="Standard Processing for SodaLog"
        type="Process"
        programName="SingleProcessStep"
        programType="com"
        active="true">
    </Step>

    <Step id="DatabaseLoad"
        description="Database Load for SodaLog"
        type="Process"
        programName="DBLoad"
        programType="com"
        active="true">
    </Step>

    <Step id="Cleanup"
        description="Cleanup SodaLog"
        type="Process"
        programName="Cleanup"
        programType="net"
        active="true">
        <Parameters>
            <Parameter DaysToRetainFiles="45"/>
        </Parameters>
    </Step>
</Steps>
</Process>

```

For a description of the `Parameter` element attributes that are specific to the Universal collector (that is, the parameters provided for the `ConvertToCSR` step), see [Table 14-8](#) on page 14-27. These parameters are used by the conversion script, `Universal.wsf`.

For a description of all other elements and attributes in the process, see [Creating Job Files](#) on page 2-25.

Parameter	Description/Values
LogDate	<p>The log date specifies that appears in the initial CSR file name. This is the CSR file that is processed by the Scan program. The start and end dates that appear in the CSR file records are defined by the conversion definition (see <i>Date/Time Tab</i> on page 14-21).</p> <p>For more information about using a log date, including valid log date values, see <i>Specifying Log Dates for Collection</i> on page 2-3</p>
RetentionFlag	This parameter is for future use.
Feed	<p>The name of the server that contains the log file that you want to collect.</p> <p>A subfolder with the same name as the server is automatically created in the process definition folder (see the <i>OutputFolder</i> parameter). This subfolder is used to store the initial CSR file that is created by the collector (see <i>Feed Subfolder</i> on page 2-13). This is the CSR file that is processed by the Scan program.</p>
OutputFolder	<p>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.</p> <p>The output folder is defined by the <i>Process id</i> attribute in the job file. For example, if the <i>Process id</i>="Unisys", the output folder is Unisys.</p>
ConvEngDefName	The location of the conversion definition file.
InputFileName	The location of the usage metering file to be processed.

**Table 14-8 • Universal.wsf Parameters**

## Running the Universal Collector

Use CIMS Job Runner to run the Universal collector as described in *Running CIMS Job Runner* on page 2-93.

## Example Files

An example usage metering file, conversion definition, and output CSR file are in `Processes\Universal` where the folder `Processes` can be in any location (see [About the Processes Folder](#) on page 2-12). These files are named `SodaLog.txt`, `SodaLogDef.txt`, and `CurrentCSR.txt`, respectively.

The following sections describe each of these files.

### Log File—SodaLog.txt

The file `SodaLog.txt` is a log file for the fictional “ACME Soda Tracker” program. This program monitors the refrigerator in the break room and generates a log entry every time someone removes a soda can. Each entry records the date, time, name of the person removing the soda, and the number of soda cans removed. The log file contains the following data:

---

01062004	05:27	MARY	1
01062004	07:13	RON	1
01062004	10:20	BERT	1
01062004	11:01	JANICE	1
01062004	12:23	JANICE	1
01062004	12:34	RANDY	1
01062004	16:02	TONY	1
01062004	17:37	JERRY	1

---

## Conversion Definition File—SodaLogDef.txt

The following sections describe the SodaLogDef.txt as viewed in CIMS Conversion Builder and Notepad.

### Conversion Definition Viewed in CIMS Conversion Builder

The information contained in the SodaLogDef.txt file is grouped by tabs in CIMS Conversion Builder as shown in this section. Note that the examples in this section reflect the options set in the SodaLogDef.txt file and that not all of the options available on the tabs are described. For a detailed description of each tab option, see [Creating a Conversion Definition Using CIMS Conversion Builder](#) on page 14-3.

#### Input Tab

The **Input** tab defines the description (optional), file type, and format for the SodaLog.txt file. Note that the input file path and name is defined in the SodaLogDef.txt definition file for example purposes only. If you are running CIMS Conversion Engine from the Universal collector, the file path and name is defined in the collector's job file (see [Creating a Job File](#) on page 14-25).

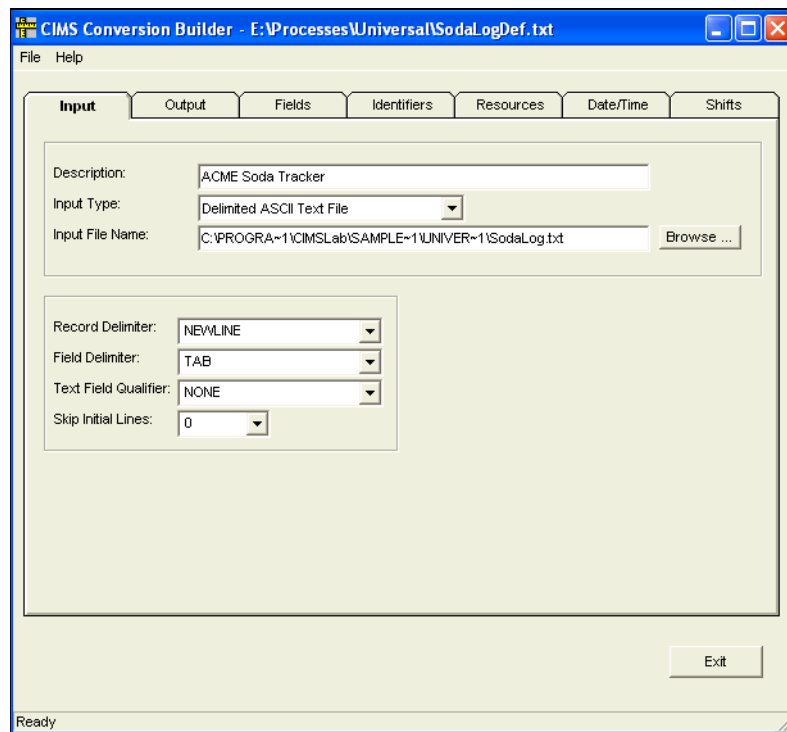


Figure 14-1 • Input Tab

### Output Tab

A CSR file named CurrentCSR.txt will be generated and stored in the Universal folder. The header SAMPLE will appear in the CSR records. Note that the output file path and name defined in the SodaLogDef.txt definition file for example purposes only. If you are running CIMS Conversion Engine from the Universal collector, the file path and name is defined in the collector's job file (see *Creating a Job File* on page 14-25).

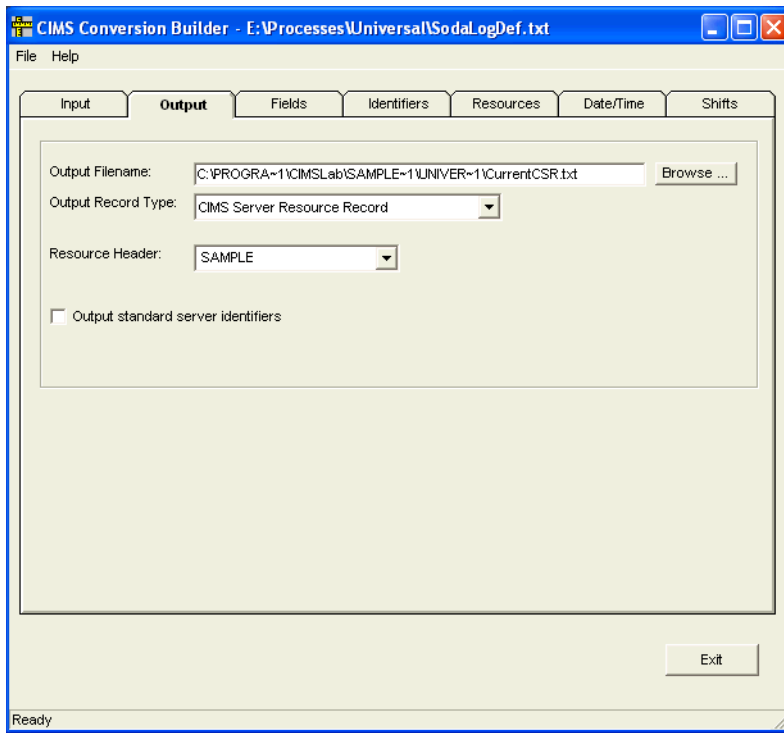


Figure 14-2 • Output Tab



### Fields Tab

This log has four fields (the date, time, user name, and number of sodas removed). The field names DATE, TIME, USER, and SODA are assigned to the fields respectively. Note that the formats for the DATE and TIME fields are declared in the Type field (see page 14-12).

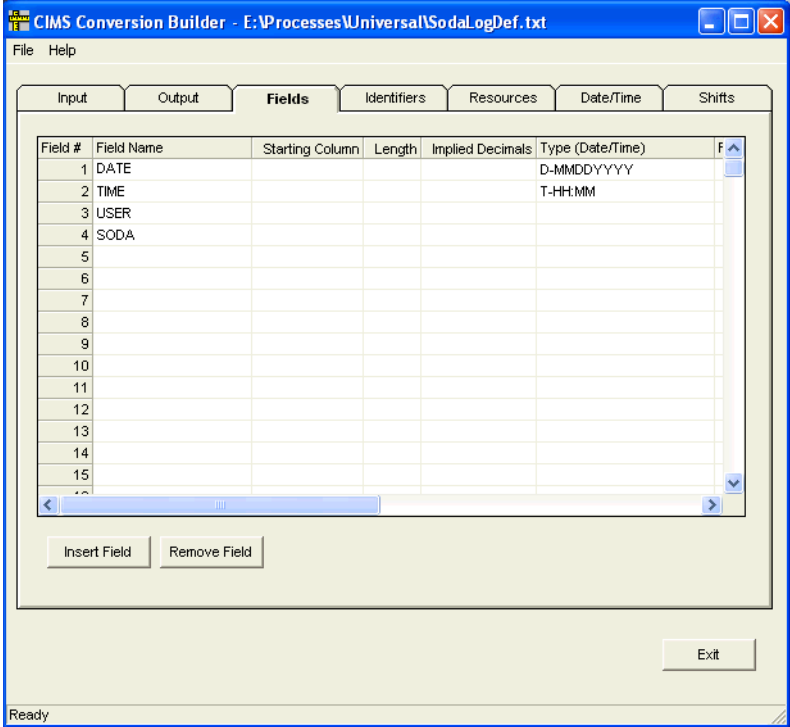


Figure 14-3 • Fields Tab

### Identifiers Tab

The identifiers in the log file are contained in the DATE, TIME, and USER fields.

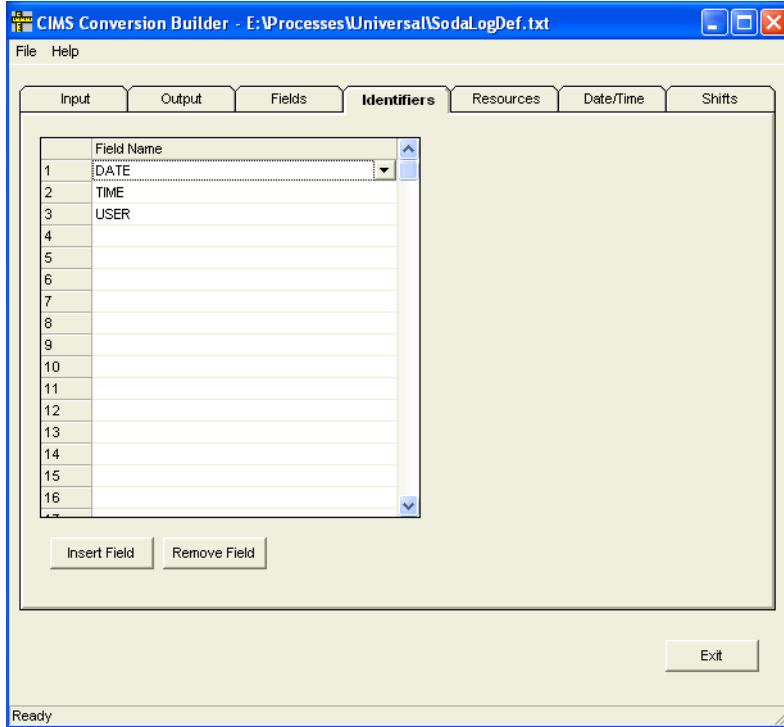


Figure 14-4 • Identifiers Tab

### Resources Tab

The field SODA represents the resources being consumed. A rate code named EMPBEV (for employee beverage) has been assigned to identify the resource. This rate code appears in the invoices and other reports generated by CIMS Server.

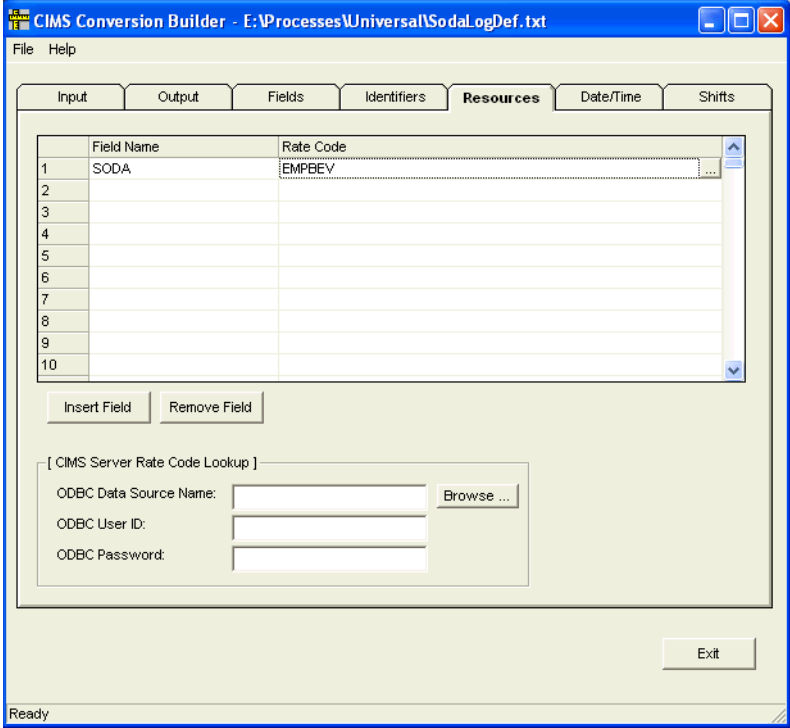


Figure 14-5 • Resources Tab

### Date/Time Tab

The system date will appear as the start/end date and time in the CSR records.

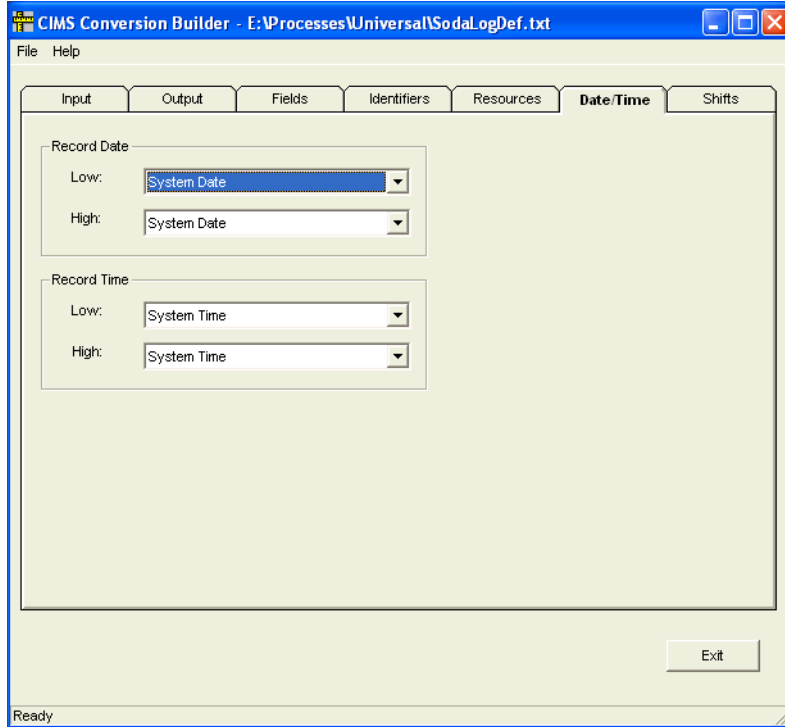


Figure 14-6 • Date/Time Tab

Shifts Tab (Optional)

Rate shifts allow you to set different rates based on the time of day. For example, if employees are charged for sodas, the rate might differ depending on the shift. In this example, the TIME field is entered in the Shift Field box. This specifies that shift character that appears in the output file records is determined by the time in the TIME field. If None is selected in the Shift Field box, the shift character is determined by the start date in the output file record (see Date/Time Tab on page 14-21).

Shifts are represented by a numeric value 1-9. This example indicates that on all records for Monday through Friday, 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.

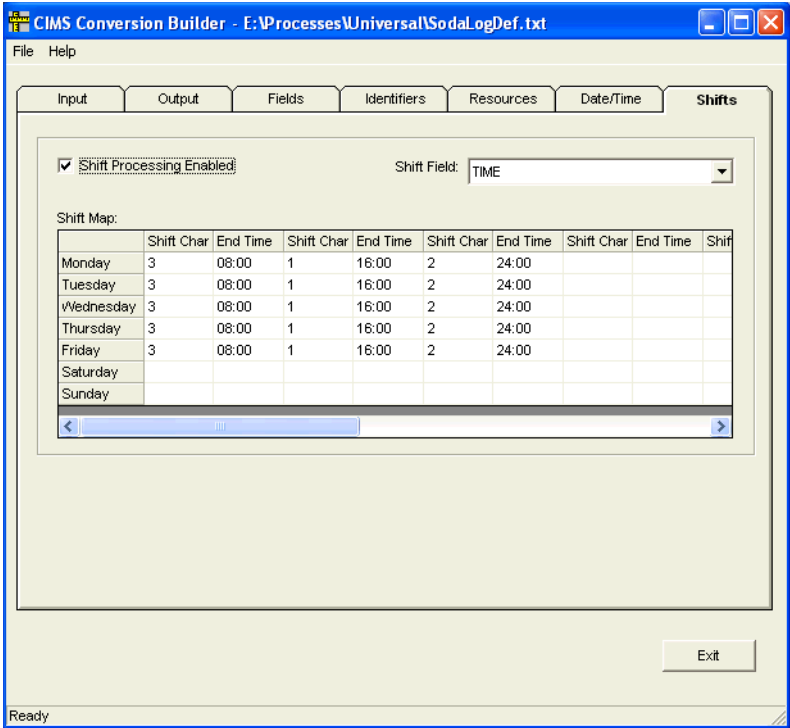


Figure 14-7 • Shifts Tab

## Conversion Definition Viewed in Notepad

The file SodaLogDef.txt contains ASCII text in the same format as a Windows .INI file. Each line in the file holds a single statement and must end with the CRLF pair.

```
[Control]
Description=ACME Soda Tracker
InputFile=C:\PROGRA~1\CIMSLab\SAMPLE~1\UNIVER~1\SodaLog.txt
OutputFile=C:\PROGRA~1\CIMSLab\SAMPLE~1\UNIVER~1\CurrentCSR.txt
OutRecType=CBS
ProcessType=DELIMITED
Delimiter=TAB
RecDelimiter=NEWLINE
InitialSkipLineCnt=0
TextQualifier=NONE
RecDateLo=SYSTEM
ShiftField=TIME
ShiftMON=DEFINE 3 0800 1 1600 2 2400
ShiftTUE=DEFINE 3 0800 1 1600 2 2400
ShiftWED=DEFINE 3 0800 1 1600 2 2400
ShiftTHU=DEFINE 3 0800 1 1600 2 2400
ShiftFRI=DEFINE 3 0800 1 1600 2 2400
ShiftsEnabled=YES
UnivHdr=SAMPLE
WriteStandardServerIdentifiers=NO

[Layout]
Field1=DATE TYPE(D-MMDDYYYY)
Field2=TIME TYPE(T-HH:MM)
Field3=USER
Field4=SODA
IDField1=DATE
IDField2=TIME
IDField3=USER
RSField1=SODA RATECODE(EMPBEV)
```

The conversion definition is divided into two sections: [CONTROL] and [LAYOUT]. The [CONTROL] section includes option statements that guide the processing performed by CIMS Conversion Engine. The [LAYOUT] section describes the data fields within the log file. For a description of the statements and values used in the conversion definition, see [Creating a Conversion Definition Using CIMS Conversion Builder](#) on page 14-3.

Comments may be added on any line in the conversion definition. The line must start with a semicolon (;) in column 1. For example:

```
; This is a comment line
```

## Output File—CurrentCSR.txt

If you ran the Universal collector on April 19, 2005, the output CSR file, CurrentCSR.txt, created from the SodaLog.txt log would contain 20050419 in the start and end date fields and the system time in the start and end time fields as shown in the following example:

```
SAMPLE,20050419,20050419,10:52:27,10:52:27,3,3,DATE,"01062004",TIME,"05:27",USER,"MARY",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,3,3,DATE,"01062004",TIME,"07:13",USER,"RON",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,1,3,DATE,"01062004",TIME,"10:20",USER,"BERT",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,1,3,DATE,"01062004",TIME,"11:01",USER,"JANICE",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,1,3,DATE,"01062004",TIME,"12:23",USER,"JANICE",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,1,3,DATE,"01062004",TIME,"12:34",USER,"RANDY",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,2,3,DATE,"01062004",TIME,"16:02",USER,"TONY",1,EMPBEV,1
SAMPLE,20050419,20050419,10:52:27,10:52:27,2,3,DATE,"01062004",TIME,"17:37",USER,"JERRY",1,EMPBEV,1
```





---

## Contacting Technical Support

The CIMS Lab Technical Support department is here to answer your questions on any aspect of CIMS Lab products.

CIMS Lab technical support can be reached in the following ways:

- **Telephone:** (800) 283-4267 in USA and Canada; 916-783-8525 International
- **Email:** [support@cimslab.com](mailto:support@cimslab.com)
- **Fax request:** (916) 783-2090

International customers may contact one of our authorized international partners. Contact CIMS Lab for more information.

In addition, customers may visit the Customer Area on our Web site for product downloads, updates, technical documentation, and password information. We are on the Web at <http://www.cimslab.com>.





---

# CIMS Aggregation Engine API

<b>About CIMS Aggregation Engine</b> .....	<b>A-2</b>
About the Aggregation Algorithm .....	A-2
Processing the Log File .....	A-3
Processing the Work File .....	A-3
<b>CIMS Aggregation Interfaces</b> .....	<b>A-3</b>
<b>TypedEngine and ScriptingEngine Interfaces</b> .....	<b>A-4</b>
About Specifying Dates and Times .....	A-4
TypedEngine and ScriptingEngine Interface Properties .....	A-5
TypedEngine and ScriptingEngine Interface Methods .....	A-12
<b>ExceptionFile Interface</b> .....	<b>A-17</b>
ExceptionFile Interface Properties .....	A-17
ExceptionFileInterface Method .....	A-18

## About CIMS Aggregation Engine

CIMS Aggregation Engine (CIMSAggregation.dll) is a COM object that aggregates the records in the log file by identifier values. CIMS Aggregation Engine provides methods for uniquely identifying an aggregate within a log file, summarizing and storing information about the aggregate, and writing the aggregate information to a CIMS Server Resource (CSR) file.

Aggregation reduces the amount of data from a log file that must be processed by CIMS Processing Engine, thus reducing processing time. This is especially beneficial for log files that are created daily and contain gigabytes of data.

CIMS Aggregation Engine is designed to be called by compiled code or scripts that pass lists of identifier names, identifier values, rate codes, and resource values from the log file, as well as optional start and end dates. CIMS Aggregation Engine then generates an aggregation key for each unique set of matching identifier values. For example, three aggregation keys, BERTACME1, JANICEACME1, and RANDYACME1 would be generated for the following log file records. The log file is generated by a fictional software program, "ACME Soda Tracker".

```
ACMESODA,20051031,20051031,11:02:43,,2,User,BERT,Machine,ACME1,1,SODA,1
ACMESODA,20051031,20051031,11:02:57,,2,User,JANICE,Machine,ACME1,1,SODA,1
ACMESODA,20051031,20051031,12:05:34,,2,User,JANICE,Machine,ACME1,1,SODA,1
ACMESODA,20051031,20051031,12:10:05,,2,User,RANDY,Machine,ACME1,1,SODA,1
```

Once an aggregation key is created, resource values passed to CIMS Aggregation Engine are matched to the key and added to the existing aggregated resource values associated with the key. For example, in the preceding log file, the second and third records share the same aggregation key. CIMS Aggregation Engine would aggregate these records to produce a resource value of 2 for the rate code SODA.

After all log file records have been passed to CIMS Aggregation Engine, the engine writes a CSR file.

## About the Aggregation Algorithm

The base aggregation algorithm used by CIMS Aggregation Engine is the Repeated Scanning algorithm<sup>1</sup>. The algorithm maintains as many aggregates in memory as possible. When no more aggregates can fit into memory, new aggregates are written to a work file. Only relevant information from each record, such as identifier and resource values, are written to the work file.

<sup>1</sup>. See *Grouping and Duplication Elimination: Benefits of Early Aggregation*, Microsoft Corporation, January 1997, <http://www.research.microsoft.com/~palarson>.

## Processing the Log File

CIMS Aggregation Engine continues to read the log file until it reaches the end of the file. Existing or new aggregates found in the log file are updated in main memory. When CIMS Aggregation Engine reaches the end of the file, the aggregates stored in memory are written out to the CSR file and cleared from memory. If a work file was written, a loop is entered to process the work file repeatedly until it is no longer required (see *Processing the Work File*).

## Processing the Work File

The base aggregation algorithm is also used to process the work file with the exception that CIMS Aggregation Engine handles all Input/Output (I/O). The number of passes required to process the work file is the total number of aggregate entries in the input log file divided by the number of aggregate entries that will fit in memory. It is expected that the number of aggregate entries will be low compared to the number of records in the input log file.

If the number of work file passes is high, the speed of aggregation is reduced because each generation and subsequent processing of the work file results in additional I/O, which is slower than main memory. There is an extension to the algorithm that hash partitions the single work file into multiple work files. By applying a hash function to the aggregation key, records belonging to the same aggregate are grouped together in a separate work files. The work files are then processed based on size, smallest file first. It is assumed that smaller work files will generate fewer future work files, thereby reducing overall work file data to be processed.

## CIMSAggregation Interfaces

CIMSAggregation uses the following interfaces:

- **TypedEngine.** This strongly-typed interface is used by programming languages that support strong types.
- **ScriptingEngine.** This weakly-typed (variant) interface is used primarily by scripting applications as scripting is based on a weakly-typed system. This interface delegates its calls to an instance of TypedEngine.
- **ExceptionFile.** This interface produces exception files containing unprocessed records.

The properties and methods for each interface are described in the following sections.

## TypedEngine and ScriptingEngine Interfaces

Except where noted, the properties and methods described in this section are contained in both the `TypedEngine` and `ScriptingEngine` interfaces. However, in the `ScriptingEngine` interface, all types are passed and returned as variants.

### About Specifying Dates and Times

When using the `TypedEngine` or `ScriptingEngine` interface, there are three ways to specify the dates and times that appear in the CSR file:

- The `DateKeyword` property. This property overrides the `DateStart` and `EndDate` properties and the `AddEntry` or `AddEntries` method date parameters. For this property, the start time is 00:00:00 and the end time is 23:59:59.
- The `DateStart` and `DateEnd` properties. If there is no `DateKeyword` property, the `DateStart` and `DateEnd` properties override the date parameters of the `AddEntry` or `AddEntries` method. For these properties, a time can be specified as part of the date.
- The date parameters specified by the `AddEntry` or `AddEntries` method. If there is no `DateKeyword` property or `DateStart` and `DateEnd` properties, these methods are used. For these parameters, a time can be specified as part of the date.

If none of the preceding properties or parameters are specified, the start time is 00:00:00 and the end time is 23:59:59.

## TypedEngine and ScriptingEngine Interface Properties

### AggregationList

Returns aggregated records in an array rather than writing them to a CSR file. The aggregate data can then be modified if needed.

Note that CIMS Aggregation engine does not process modified records. To write modified records to a CSR file, use a script. CIMS Lab provides the class `CSRWriter` in the `CIMSLib.wsf` script to write CSR files.

### Syntax

*object*.AggregationList

### Parameters

None.

### Comments

To use this property, the aggregates must fit in memory.

### Example

Retrieve the array:

```
Dim List
List = AggregationEngineObject.AggregationList
```

The two-dimensional array is returned in the same order as the CIMS Server Resource Record. For example, entry 0 in the array might appear as follows:

AggList (0, 0) = DEMOARRY	(Resource Header [String])
AggList (0, 1) = 4/1/2003	(Start Date/Time [Date])
AggList (0, 2) = 4/30/2003 11:59:59 PM	(End Date/Time [Date])
AggList (0, 3) = 3	(Number of Identifiers [Long])
AggList (0, 4) = ServerId	(Identifier Name [String])
AggList (0, 5) = Server#000003	(Identifier Value [String])
AggList (0, 6) = UserId	(Identifier Name [String])
AggList (0, 7) = User#000003	(Identifier Value [String])
AggList (0, 8) = FileNumber	(Identifier Name [String])
AggList (0, 9) = FileNum1	(Identifier Value [String])
AggList (0, 10) = 2	(Number of Resources [Long])
AggList (0, 11) = RES1	(Rate Code [String])
AggList (0, 12) = 6	(Resource Value [String])
AggList (0, 13) = RES2	(Rate Code [String])
AggList (0, 14) = 0.9	(Resource Value [String])

#### **DataValidation**

Returns or sets a Boolean value that indicates whether the incoming data should be verified. Verification includes scanning all input for invalid character data.

#### **Syntax**

```
object.DataValidation [=value]
```

#### **Parameter**

*Value*

A Boolean value that indicates whether incoming data should be checked.

#### **Comments**

The default value is `False`, data should not be verified. Verifying incoming data may slow down the aggregation process.

If the value is set to `True`:

- The `TypedEngine` interface makes the following checks:
  - The number of identifiers passed to the `AddEntry` or `AddEntries` method must match the number of identifiers declared by the `DefineIdentifier` method.
  - The number of resources passed to the `AddEntry` or `AddEntries` method must match the number of resources declared by `DefineResource` method.
- The `ScriptingEngine` interface makes the following checks:
  - `StartDate` and `EndDate` parameters passed to the `AddEntry` or `AddEntries` method are checked to ensure that they are valid dates.
  - Resource values passed to the `AddEntry` or `AddEntries` method are checked to ensure that they are numeric values.



## DateAggregation

Returns or sets a the date field to aggregate on.

### Syntax

*object*.DateAggregation [=value]

### Parameter

#### Value

A value specifying the date fields to aggregate on. Valid value is None, StartDate, EndDate, or Both. The enumeration values are:

- EDateAggregation\_None = 1
- EDateAggregation\_StartDate = 2
- EDateAggregation\_EndDate = 3
- EDateAggregation\_Both = 4

### Comments

The default value is to aggregate on both date fields, EDateAggregation\_Both.

If EDateAggregation\_None or EDateAggregation\_Both is specified, the CIMS Server Resource Record will contain a minimum of the start date value and the maximum of the end date value.

If EDateAggregation\_StartDate is specified, the CIMS Server Resource Record start date/time fields will contain the minimum of the start date value. The CIMS Server Resource Record end date/time fields will contain the end date/time from the first record.

If EDateAggregation\_EndDate is specified, the CIMS Server Resource Record end date/time fields will contain the maximum of the end date value. The resource record start date/time fields will contain the start/time date from the first record.

To perform date aggregation, the date parameters must be specified in the AddEntry or AddEntries method.

## DateEnd

Returns or sets a default date value that specifies the ending date field to be written to the records in the CSR file.

### Syntax

*object*.DateEnd [=value]

### Parameter

#### Value

A date value specifying the date end value of the CIMS Server Resource Record.

### Comments

This property is overridden if the DateKeyword property is specified.

## **DateKeyword**

Returns or sets a string value that specifies a keyword that determines the date range to use for date field values to be written to the records in the CSR file.

### **Syntax**

*object.DateKeyword* [=value]

### **Parameter**

*Value*

A pre-defined keyword value. Valid values are:

- **"\*\*RNDATE"** or **"\*\*CURDAY"**—Sets date range based on the run date.
- **"\*\*CURDAY"**—Sets date range based on the run date.
- **"\*\*CURWEK"**—Sets date range based on the run week (Sun–Sat).
- **"\*\*CURMON"**—Sets date range based on the run month.
- **"\*\*PREDAY"**—Sets date range based on the run date, less one day.
- **"\*\*PREWEK"**—Sets date range based on the previous week (Sun–Sat).
- **"\*\*PREMON"**—Sets date range based on the previous month.

### **Comments**

This property overrides:

- The date parameters specified in the `AddEntry` or `AddEntries` method.
- The date specified by the `DateStart` and `DateEnd` properties.

## **DateStart**

Returns or sets a default date value that specifies the starting date field to be written to the records in the CSR file.

### **Syntax**

*object.DateStart* [=value]

### **Parameter**

*Value*

A date value specifying the date start value of the CIMS Server Resource Record.

### **Comments**

This property is overridden if the `DateKeyword` property is specified.

## DebugMessage

Returns a string value that contains detailed internal counters about the aggregation run.

### Syntax

*object*.DebugMessage

### Parameters

None.

### Comments

None.

## LastErrorMessage

Returns a string value description of the error message generated by the last method or property call.

### Syntax

*object*.LastErrorMessage

### Parameters

None.

### Comments

If no errors are generated by the last method or property call, an empty string is returned.

## MemoryMinimum

Returns or sets an integer value that specifies the minimum amount of memory in megabytes that CIMS Aggregation Engine will use to store aggregates.

### Syntax

*object*.MemoryMinimum [=value]

### Parameter

*Value*

An integer value specifying the minimum amount of memory used to store aggregates.

### Comments

CIMS Aggregation Engine will use the amount of memory specified by the minimum memory value even if the amount of physical memory available is less than this value. This property is useful when other processes consume all available physical memory. By specifying a minimum, CIMS Aggregation Engine might be able to force the release of some physical memory for its use.

The aggregation engine will request that operating system set the process working set size to be in the range set by the `MemoryMinimum` and `MemoryMaximum` properties. The process working set is the set of memory pages currently visible to the process in physical memory. These pages are resident and available for use without triggering a page fault.

## **MemoryMaximum**

Returns or sets an integer value that specifies the maximum amount of memory in megabytes that CIMS Aggregation Engine will use to store aggregates.

### **Syntax**

*object*.MemoryMaximum [=value]

### **Parameter**

*Value*

An integer value specifying the maximum amount of memory used to store aggregates.

### **Comments**

The aggregation engine will request that operating system set the process working set size to be in the range set by the `MemoryMinimum` and `MemoryMaximum` properties. The process working set is the set of memory pages currently visible to the process in physical memory. These pages are resident and available for use without triggering a page fault.

## **OutputFileName**

Returns or sets a string value that specifies the output file name of the CSR file to be written.

### **Syntax**

*object*.OutputFileName [=value]

### **Parameter**

*Value*

A full path and file name that determines where the CSR file will be written.

### **Comments**

The parameter must be specified. There is no default.

## **ResultsMessage**

Returns a string value that contains detailed internal counters about the aggregation run.

### **Syntax**

*object*.ResultsMessage

### **Parameters**

None.

### **Comments**

None.

**WorkFilePath**

Returns or sets a string that specifies a complete file system path where the work files, if required, will be written.

**Syntax**

*object*.WorkFilePath [=value]

**Parameter**

*Value*

A string specifying the complete file system path where the work files will be written.

**Comments**

The default is to use the path specified by the TEMP environment variable. If the TEMP environment variable is not defined, the current directory is used.

Work files are not always generated. Work files are generated when all of the aggregates will not fit into memory.

## TypedEngine and ScriptingEngine Interface Methods

### AddEntry

Adds a list of identifier values and resource values to an aggregate.

#### Syntax

```
object.AddEntry(ByRef IdentifierValueList() As String, _  
                ByRef ResourceValueList() As String, _  
                Optional ByVal DateStart As Date, _  
                Optional ByVal DateEnd As Date) As Long
```

#### Parameters

##### *IdentifierValueList*

Provides a list of identifier value strings (it cannot be an empty a list). The number of entries must match the number of entries specified in the `DefineIdentifier` method.

##### *ResourceValueList*

Provides a list of numeric resource values (it cannot be an empty a list). The number of entries must match the number of entries specified in the `DefineResource` method.

##### *DateStart*

An optional parameter that specifies the starting date for this entry. If no entry is specified, the default specified with the `DateStart` or `DateKeyword` property is used.

##### *DateEnd*

An optional parameter that specifies the ending date for this entry. If no entry is specified, the default specified with the `DateEnd` or `DateKeyword` property is used.

#### Return Value

Returns a CIMS result code indicating whether the entry specified was added successfully. The CIMS result codes are:

- Successful = 0
- Warning = 8
- Error = 16

#### Comments

The `DateStart` and `DateEnd` parameters are overridden if:

- The `DateKeyword` property is specified.
- The `DateStart` and `DateEnd` properties are specified.

To perform date aggregation, the `DateStart` and `DateEnd` parameter values must be specified.

The identifier value list is matched in the same order as identifier names are defined. The resource value list is matched in the same order as rate codes are defined.

## AddEntries

Batches several calls to the `AddEntry` method into a single call resulting in lower processing overhead.

### Syntax

```
object.AddEntries(ByVal NumberOfEntries As Variant, _
                 ByRef IdentifierValueList() As Variant, _
                 ByRef ResourceValueList() As Variant, _
                 Optional ByRef DateStartList As Variant, _
                 Optional ByRef DateEndList As Variant) _
                 As Long
```

### Parameters

*NumberOfEntries*

Specifies the number of valid entries contained in the identifier value string lists.

*IdentifierValueList*

Provides a list of identifier value strings (it cannot be an empty a list). The number of identifier values must match the number of entries specified in the `DefineIdentifier` method.

The array must be declared with the number of identifier values first, followed by the number of entries in the list. For example, if there are 1000 entries each with 3 identifier values, the array is declared in VBScript as (2, 999). (Arrays in VBScript begin counting at 0).

*ResourceValueList*

Provides a list of numeric resource values (it cannot be an empty a list). The number of resource values must match the number of entries specified in the `DefineResource` method.

The array must be declared with the number of resource values first, followed by the number of entries in the list. For example, if there are 1000 entries each with 3 resource values, the array is declared in VBScript as (2, 999).

*DateStartList*

An optional parameter that specifies the starting date list. If no entry is specified, the default specified with the `DateStart` or `DateKeyword` property is used. If a list is specified, all entries in the list must contain a valid date.

*DateEndList*

An optional parameter that specifies the ending date list. If no entry is specified, the default specified with the `DateEnd` or `DateKeyword` property is used. If a list is specified, all entries in the list must contain a valid date.

#### **Return Value**

Returns a CIMS result code indicating whether all of the entries specified were added successfully. The CIMS result codes are:

- Successful = 0
- Warning = 8
- Error = 16

#### **Comments**

This method is currently implemented only in the `ScriptingEngine` interface.

The `DateStartList` and `DateEndList` parameters are overridden if:

- The `DateKeyword` property is specified.
- The `DateStart` and `DateEnd` properties are specified.

To perform date aggregation, the `DateStartList` and `DateEndList` parameter values must be specified.

The identifier value list is matched in the same order as identifier names are defined. The resource value list is matched in the same order as rate codes are defined.

#### **ClearIdentifierList**

Clears the internal list of identifier names.

#### **Syntax**

```
object.ClearIdentifierList()
```

#### **Parameters**

None.

#### **Comments**

None.

#### **ClearResourceList**

Clears the internal list of rate codes.

#### **Syntax**

```
object.ClearResourceList()
```

#### **Parameters**

None.

#### **Comments**

None.



## DefineIdentifier

Adds an identifier name to an internal list of identifier names.

### Syntax

```
object.DefineIdentifier(ByVal IdentifierName As String)
```

### Parameter

*IdentifierName*

Provides a string value containing an identifier name.

### Comments

Identifiers names must be defined in the same order that the identifier values appear in the `AddEntry` or `AddEntries` method.

There must be at least one identifier name defined.

## DefineResource

Adds a rate code to an internal list of rate codes.

### Syntax

```
object.DefineResource(ByVal RateCode As String,  
                    Optional ByVal ResourceConversionFactor As Double  
                    Optional ByVal DecimalPositions As Long)
```

### Parameters

*RateCode*

Provides a string value containing a rate code.

*ResourceConversionFactor*

An optional parameter that divides the incoming resource values passed to the `AddEntry` or `AddEntries` method by a double value. The default value is 1. This is an optional parameter.

*DecimalPositions*

An optional parameter that specifies the number of decimal digits that resource values are rounded to. Zero rounds to a whole number. By default, the values are not rounded.

Rounding is based on 5. For example, a resource value of 3.5 rounds to 4 if 0 is specified for the decimal digits. A value of 5.53 rounds to 5.5 if a decimal digit of 1 is specified.

### Comments

Rate codes must be defined in the same order that the resource values appear in the `AddEntry` or `AddEntries` method.

There must be at least one rate code defined.

If a resource conversion factor of 1 is specified, then no division of resource values takes place.

## **DefineResourceRecordHeader**

Specifies the resource record header that the records in the CSR file should use.

### **Syntax**

```
object.DefineResourceRecordHeader(ByVal ResourceRecordHeader As String)
```

### **Parameter**

*ResourceRecordHeader*

Provides a string value containing the resource record header to be used for records generated by the `AddEntry` or `AddEntries` method.

### **Comments**

This method can be set once for all records, called once for each record, or called as needed. The default value is `NONE`.

## **Initialize**

Initializes the aggregation object.

### **Syntax**

```
object.Initialize(Optional ByVal MaxEntries as Long) As Boolean
```

### **Parameter**

*MaxEntries*

An optional parameter that specifies how many aggregates to store in memory. The default is to store as many aggregates as will fit in memory.

### **Return Value.**

Returns `True` if initialization is successful. Returns `False` if otherwise.

### **Comments**

This method should be the first call made to CIMS Aggregation Engine. It resets all properties to their default values and resets the internal state of the object.

## **WriteResourceFile**

Releases all aggregation records to the CSR file.

### **Syntax**

```
object.WriteResourceFile() As Boolean
```

### **Parameters.**

None.

### **Return Value.**

Returns `True` if the CSR file was written successfully. Returns `False` if otherwise.

### **Comments**

This method must be called for the CSR file to be written. If all aggregates do not fit into memory, this method initiates work file processing. When the method returns, full aggregation of the input file has been completed and the CSR file has been written.

# ExceptionFile Interface

## ExceptionFile Interface Properties

### ExceptionCount

Returns a count of the number of exception records written so far.

#### Syntax

*object*.ExceptionCount

#### Parameters

None.

#### Comments

None.

### FileName

Returns or sets the name of the exception file.

#### Syntax

*object*.FileName [=value]

#### Parameters

*Value*

A string value specifying the full path and file name of the exception file.

#### Comments

The default file name is CIMSExceptionFile.txt.

### MaxExceptions

Returns or sets the maximum number of exception entries.

#### Syntax

*object*.MaxExceptions [=value]

#### Parameters

*Value*

A long value specifying the maximum number of entries that can be written to the exception file.

#### Comments

To allow an unlimited number of exception entries, set this property to -1.

The default is to allow an unlimited number of exception entries.

Once the maximum number of exceptions has been reached, no more entries are written to the exception file.

### **MaxExceptionsReached**

Returns a Boolean value indicating whether the maximum number of exceptions generated by the `AddException` method exceed the number specified by the `MaxException` property.

#### **Syntax**

*object*.MaxExceptionsReached

#### **Parameters**

None.

#### **Comments**

If an unlimited number of exception entries is allowed (the default), then the return value is always `False`.

## **ExceptionFileInterface Method**

### **AddException**

Adds an exception record to an exception file.

#### **Syntax**

*object*.AddException(ByVal *Value* as Long) As String

#### **Parameter**

*Value*

A string value that contains the source record that could not be processed.

#### **Return Value**

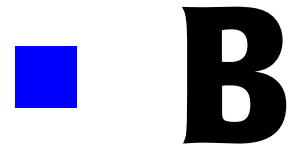
Returns `True` if the source record string could be added to the exception file. Returns `False` otherwise.

#### **Comments**

An exception file name must be specified by setting the `FileName` property.

The CSR file is closed when the script exits or the object goes out of scope.

If no exceptions are generated, the exception file is not created.



---

# CIMS Data Sources

**About CIMS Data Sources .....B-2**  
**Creating a CIMS Data Source .....B-3**

## About CIMS Data Sources

The CIMS Data Collectors that collect data from a database or databases (SQL Server, DBSpace, Citrix, and VMware) use a CIMS Data Source ID to identify the database(s).

The configuration of the CIMS Data Source is as follows: CIMS Data Source -> ODBC Data Source -> database. Where the CIMS Data Source points to an ODBC Data Source that points to a database.

CIMS Data Sources are created and maintained in CIMS Server Administrator as described in *Creating a CIMS Data Source* on page B-3. The requirements for setting up the CIMS Data Source differ depending on the type of database that the collector is collecting data from as shown in *Table 2-1*.

Collector	Requirement
SQL Server collector	<p>This collector gathers data from SQL Server databases.</p> <p>The CIMS Data Source must point to the database that contains the CIMSSp_SQLServer2000Trace stored procedure. The collector will then collect data from all databases in the same instance.</p>
DBSpace collector	<p>This collector gathers data from SQL Server or Sybase databases.</p> <p>The CIMS Data Source can point to any database in an instance. The collector will then collect data from all databases in the instance.</p>
Citrix and VMware collectors	<p>These collectors collect data from a SQL Server or Oracle database depending on the type of databases used for the Resource Manager summary database (Citrix) or VirtualCenter database (VMware).</p> <p>The CIMS Data Source must point directly to the database that you want to collect data from.</p>

**Table 2-1 • CIMS Data Source Requirements by Collector**

## Creating a CIMS Data Source

If you are using the SQL Server, DBSpace, Citrix, or VMware collector to collect data from a SQL Server database or databases, follow the instructions in [To create a CIMS Data Source and ODBC Data Source for a SQL Server Database](#).

If you are using the Citrix or VMware collector to collect data from an Oracle database, follow the instructions in [To create a CIMS Data Source and ODBC Data Source for an Oracle Database](#): on page B-5.

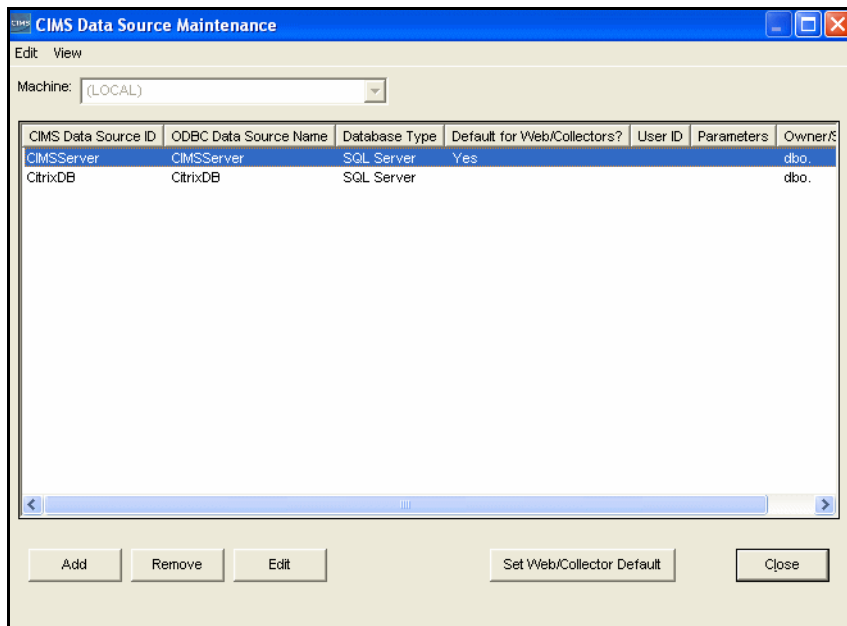
### **To create a CIMS Data Source and ODBC Data Source for a SQL Server Database:**

- 1** In the CIMS Server Administrator main window, expand **System Administration**, and then click **CIMS Data Source Maintenance**.
- 2** In the CIMS Data Source Maintenance dialog box, click **Add**.
- 3** In the Add CIMS Data Source dialog box, type an ID for the CIMS Data Source, and then click **OK**. For example, if you are creating the CIMS Data Source for the Citrix Resource Manager summary database, and the database name is CitrixDB you might type CitrixDB as the ID.
- 4** In the Configure CIMS Data Source dialog box, do the following:
  - CIMS Data Source must point to an ODBC Data Source for the database. If an ODBC Data Source for the database that you want to point to *does not* appear in the **Select a System ODBC Data Source** list, click **ODBC Data Source Administrator** and continue to [Step 5](#).
  - If the ODBC Data Source that you want to point to *does* appear in the **Select a System ODBC Data Source** list, click that data source and continue to [Step 8](#).
- 5** On the ODBC Data Source Administrator **System DSN** tab, click **Add**. The Create New Data Source wizard appears.
- 6** Follow the steps provided in the wizard to point to the database. Consult your SQL Server DBA to determine the settings that you should select in the wizard. Note the following:
  - The wizard prompts you to type a name for the ODBC Data Source. For example, if the CIMS Data Source ID is Citrix, you might also type CitrixDB for the ODBC Data Source name.
  - The wizard prompts you with the following: **How should SQL Server verify the authenticity of the login ID?** Click the correct authentication method (Windows or SQL Server). If you select SQL Server authentication, type the SQL Server user ID and password in the **Login ID** and **Password** boxes.
- 7** When the wizard completes, click **Test Data Source** to make sure that the configuration was successful, and then click **OK** until the wizard and ODBC Data Source Administrator dialog boxes close.

- 8 In the Configure CIMS Data Source dialog box, do the following:
  - a In the **Select a System ODBC Data Source** list, make sure that the ODBC Data Source that you want to point to is selected.
  - b In the **User ID** and **Password** boxes, type the SQL Server user ID if you selected SQL Server authentication in **Step 6**. The password is encrypted.
  - c In the **Owner/Schema** box, set the applicable database prefix or use the default `dbo.` prefix. For more information, consult your SQL Server DBA.
  - d In the **Additional Parameters** box, type any additional parameters that are required to enable connection to the database. For more information, consult your SQL Server DBA or CIMS Lab.
- 9 Click **OK** to save the data source information and close the Configure CIMS Data Source dialog box. The data source entry appears in the CIMS Data Source Maintenance dialog box.

In example **Figure B-1**, a CIMS Data Source has been created for a Citrix Resource Manager summary database running on SQL Server. The CIMS Data Source ID is CitrixDB and the ODBC Data Source name is also CitrixDB.

In this example, the data source named CIMSServer points the CIMS Server database and is set as the default under **Default for Web/Collectors?**. This specifies that the CIMSServer database is used as the default for Web and batch reporting and for loading processed data for CIMS Server. Make sure that you do not select a non-CIMS Server database, such as the Citrix or VMware database, as the default.



**Figure B-1 • CIMS Data Source for a SQL Server Database Example**



---

**To create a CIMS Data Source and ODBC Data Source for an Oracle Database:**

---

**Note** • The following steps are required if you are using the Citrix or VMware collector to collect data from an Oracle database. If you want to collect system data from an Oracle database, these steps are not required. See [Oracle Data Collector](#) on page 4-18.

---

- 1** In the CIMS Server Administrator main window, expand **System Administration**, and then click **CIMS Data Source Maintenance**.
- 2** In the CIMS Data Source Maintenance dialog box, click **Add**.
- 3** In the Add CIMS Data Source dialog box, type an ID for the CIMS Data Source, and then click **OK**. For example, if you are creating the CIMS Data Source for the Citrix Resource Manager summary database and the database name is Citrix DB, you might type CitrixDB as the ID.
- 4** In the Configure CIMS Data Source dialog box, do the following:
  - CIMS Data Source must point to an ODBC Data Source for the database. If an ODBC Data Source for the database that you want to point to *does not* appear in the **Select a System ODBC Data Source** list, click **ODBC Data Source Administrator** and continue to [Step 5](#) on page B-5.
  - If the ODBC Data Source that you want to point to *does* appear in the **Select a System ODBC Data Source** list, click that data source and continue to [Step 9](#).
- 5** On the ODBC Data Source Administrator **System DSN** tab, click **Add**. The Create New Data Source wizard appears.
- 6** Click the Oracle driver that you want to use to set up the data source and then click **Finish**. The Oracle ODBC Driver Configuration dialog box appears.
- 7** In the Oracle ODBC Driver Configuration dialog box, do the following. Consult your Oracle DBA for assistance. The dialog box also includes a **Help** button.
  - a** In the **Data Source Name** box, type the name that you want to assign to the ODBC data source. For example, if the CIMS Data Source ID is CitrixDB, you might also type CitrixDB for the ODBC Data Source name.
  - b** In the optional **Description** box, type a description of the ODBC Data Source.
  - c** In the **TNS Service Name** box, enter the location of the Oracle database from which the ODBC driver will retrieve data. This is the same name entered in the `tnsnames.ora` file using the Oracle Net Manager.
  - d** In the optional **User ID** box, type a user ID for the Oracle database from which the ODBC drive will retrieve the data. If you do not provide the user ID in this box, you can type it when you test the connection to the database in [Step 8](#).
  - e** You can leave the remaining options in the dialog box set to the defaults or change the options as needed.

- 8** When you have set the options in the Oracle ODBC Driver Configuration dialog box, click **Test Connection** to make sure that the configuration was successful. The Oracle ODBC Driver Connect dialog box appears. Type a user ID and password for the database and click **OK**. If you entered a user ID in the **User ID** box, that user ID appears by default.
- 9** Click **OK** until you return to the Configure CIMS Data Source dialog box and do the following. Consult your Oracle DBA for assistance.
  - a** In the **Select a System ODBC Data Source** list, make sure that the ODBC Data Source that you want to point to is selected.
  - b** In the **User ID** and **Password** boxes, type the Oracle user ID and the password. The password is encrypted.
  - c** In the **Owner/Schema** box, type the name of the database schema that is used for the database. Consult your Oracle DBA for the database schema.
  - d** In the **Additional Parameters** box, type any additional parameters that are required to enable connection to the database. For more information, consult your Oracle DBA or CIMS Lab.
- 10** Click **OK** to save the data source information and close the Configure CIMS Data Source dialog box. The data source entry appears in the CIMS Data Source Maintenance dialog box.

In example [Figure B-2](#) on page B-7, a CIMS Data Source has been created for a Citrix Resource Manager summary database running on Oracle. The CIMS Data Source ID is CitrixDB and the ODBC Data Source name is also CitrixDB.

In this example, the data source named CIMSServer points the CIMS Server database and is set as the default under **Default for Web/Collectors?**. This specifies that the CIMSServer database is used as the default for Web and batch reporting and for loading processed data for CIMS Server. Make sure that you do not select a non-CIMS Server database, such as the Citrix or VMware database, as the default.

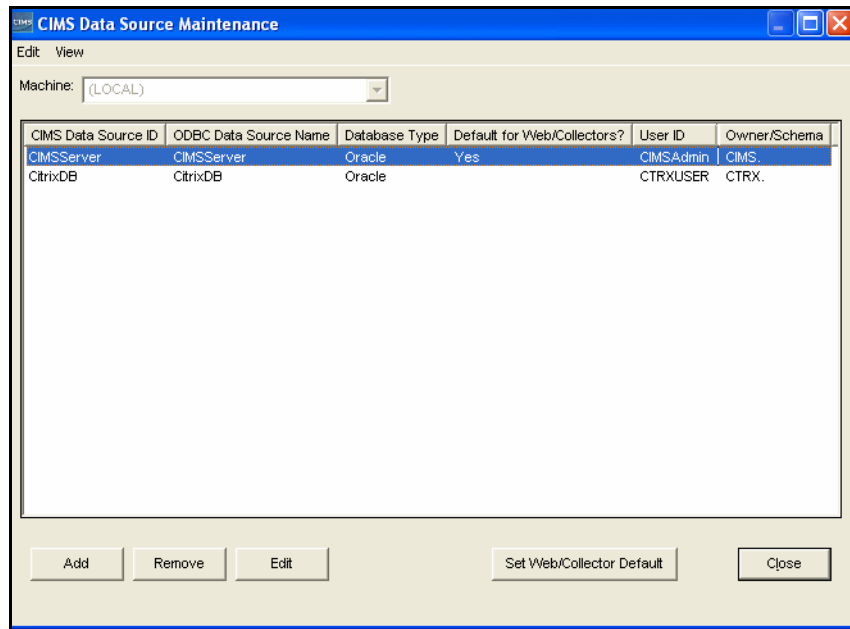


Figure B-2 • CIMS Data Source for an Oracle Database Example





---

# Glossary

**CDO** • Acronym for Collaboration Data Objects. A technology for building messaging and collaboration applications. The current version of CDO is 1.21. It is designed to simplify the creation of applications with messaging functionality, and to add messaging functionality to existing applications.

For example, CDO and Active Server Pages enable you to add scripts to a Web site to provide support for creating, sending, and receiving e-mail as well as participating in discussions and other public folder applications.

**CIMS Aggregation Engine** • CIMS Aggregation Engine is a COM object that aggregates the records within the usage metering file by identifier values and produces a CIMS Server Resource File. Because the data in the usage metering file has been aggregated, the resulting CIMS Server Resource File requires less processing time. *See also identifier.*

**CIMS Conversion Builder** • CIMS Conversion Builder is a GUI application that you can use to create definition files for the usage metering files. These definition files are fed into CIMS Conversion Engine. CIMS Conversion Builder is used only with CIMS Universal Collector. *See also CIMS Conversion Engine, CIMS Universal Data Collector, and definition file.*

**CIMS Conversion Engine** • CIMS Conversion Engine is a COM object that enables usage metering files to be processed by CIMS Server. CIMS Conversion Engine reformats the data in the files into CIMS Server Resource Files. CIMS Conversion Engine is used only with CIMS Universal Collector. *See also CIMS Universal Data Collector.*

**CIMS Job Runner** • CIMS Job Runner is a console application that runs the data collection process. CIMS Job Runner executes the jobs defined in a job file. Each job can run one or more data collectors.

**CIMS Processing Engine** • CIMS Processing Engine is composed of COM objects that process the CIMS Server Resource Files created by CIMS Aggregation Engine or CIMS Conversion Builder and load the output into the CIMS Server database.

**CIMS Server Resource (CSR) File** • The resource file that contains the data that is input into CIMS Server. 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.

**CIMS Universal Data Collector** • A universal data collection process for applications that do not have a specific CIMS Data Collector.

**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.

**CPU** • Acronym for central processing unit. The computational and control unit of a computer.

**CSR File** • *See CIMS Server Resource (CSR) File.*

**CSR+ File** • *See CIMS Server Resource Plus (CSR+) File.*

**DLL** • Acronym for dynamic-link library. A module that contain functions and data that can be used by another module (application or DLL).

**DN** • Acronym for distinguished name.

**definition file** • The definition file defines the format of the usage metering file as well as the format of the output file to be produced by CIMS Conversion Engine.

**FTP** • Acronym for File Transfer Protocol. A application-level protocol widely used for transferring both text-based and binary files to and from remote systems, especially over the Internet.

**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.

**.NET Framework** • An integral Windows component that enables building and running the next generation of software applications and Web services. It includes technologies for Web services and Web applications (ASP.NET), data access (ADO.NET), smart client applications (Windows Forms), and many others.

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

**process** • An executable application, such as Microsoft Word, or a service such as MSTask.

**process definition folder** • A folder that contains the files required to process usage data from a particular source such as a database, operating system, or application.

**rate codes** • Rate codes represent the resource units being reported (for example, CPU time, transactions processed or lines printed). Each rate code includes the value for a resource and other rate processing information.

**Secure Shell** • Sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely getting access to a remote computer. It is widely used by network administrators to control Web and other kinds of servers remotely.

**service** • A program, routine, or process that performs a specific system function to support other programs.

**UNC** • Acronym for Universal Naming Convention. A file naming system beginning with two backslashes (\\) that indicates that the resource exists on a network computer. The syntax is \\Servername\Sharename.

**usage metering file** • A file that contains usage data for an application. For example, a log file.

**UTC** • Acronym for Universal Time Coordinate. A world-wide standard for time and date. Formerly know as Greenwich Mean Time (GMT). Also referred to as Zulu time, universal time, and world time.

**VBScript** • Abbreviation for Visual Basic, Scripting Edition. A subset of the Visual Basic for Applications programming language, optimized for Web-related programming.

**Windows Script Component** • A script tool for creating COM components. Script component files are indicated by the extension .wsc. These files are XML (Extensible Markup Language) files that contain information about the COM component. *See also COM.*

**Windows Script File** • A Windows script (.wsf) file is a text document containing Extensible Markup Language (XML) code. Windows script files are not engine-specific and can contain script from any Windows Script compatible scripting engine.

**Windows Script Host (WSH)** • A language-independent scripting host for Windows Script-compatible scripting engines. WSH acts as a host for scripts—it makes objects and services available for the script and provides a set of guidelines within which the script is executed.

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







---

# Index

## A

Aggregation Engine *See* CIMS Aggregation Engine  
Apache data collector  
  about 6-33  
  identifiers and resources collected by 6-33  
  job file process example 6-33, 7-11  
  setting up 6-33 to 6-35  
Apache.wsf script, parameters for 6-34  
AS/400 data collector 3-34  
attributes  
  Default element 2-90  
  GenerateExternalFile element 2-59 to 2-60  
  Job element 2-48 to 2-51  
  Jobs element 2-46 to 2-47  
  Process element 2-52 to 2-54  
  Step element 2-55 to 2-58  
  Steps element 2-55

## B

batch program  
  for installing trace stored procedure 4-4  
BindView data collector 8-5

## C

CIMS Aggregation Engine  
  about A-2 to A-3  
  interfaces A-3  
  methods  
    AddEntries A-13  
    AddEntry A-12  
    ClearIdentifierList A-14  
    ClearResourceList A-14  
    DefineIdentifier A-15  
    DefineResource A-15

  DefineResourceRecordHeader A-16  
  ExceptionFileInterface A-18  
  Initialize A-16  
  WriteResourceFile A-16  
properties  
  AggregationList A-5  
  DataValidation A-6  
  DateAggregation A-7  
  DateEnd A-7  
  DateKeyword A-8  
  DateStart A-8  
  DebugMessage A-9  
  ExceptionCount A-17  
  FileName A-17  
  LastErrorMessage A-9  
  MaxExceptions A-17  
  MaxExceptionsReached A-18  
  MemoryMaximum A-10  
  MemoryMinimum A-9  
  OutputFileName A-10  
  ResultsMessage A-10  
  WorkFilePath A-11  
CIMS Conversion Builder  
  about 14-3  
  creating a conversion definition using 14-3 to 14-23  
    example 14-28 to 14-36  
  opening a conversion definition using 14-24  
  regular expressions, using  
    regular expressions, using in CIMS Conversion Builder 14-15  
  saving a conversion definition using 14-24

- CIMS Conversion Engine
  - about 14-2
  - and CIMS Universal Data Collector 14-2
  - running
    - from a job file (example) 14-25
    - from CIMS Conversion Builder 14-24
- CIMS Data Collectors
  - installing 2-2
  - running 2-93 to 2-95
  - system architecture, described 2-3 to 2-14
  - system specifications 2-2
  - types of 1-3
- CIMS Job Runner
  - about 2-3
  - running 2-93 to 2-95
- CIMS Universal Data Collector *See* Universal data collector
- CIMS Windows Disk data collector
  - about 7-2
  - identifiers and resources collected by 7-2
  - job file process example 7-3
  - setting up 7-3 to 7-8
  - Smart Scan, requirements for 7-8
  - XML file, structure of 7-5 to 7-8
- CIMS Windows Event Log data collector
  - about 9-2
  - Event Viewer options, setting 9-9
  - identifiers and resources collected by 9-2
  - job file process example 9-3
  - setting up 9-3 to 9-7
  - Smart Scan, requirements for 9-7
  - XML file, structure of 9-5 to 9-7
- CIMS Windows Print data collector
  - about 9-10
  - installing 9-14 to 9-15
  - job file process example
    - on central CIMS Data Collectors Server 9-20
    - on other server 9-24
  - log files
    - format of 9-17
    - identifiers and resources collected from 9-18
  - setting up 9-20 to 9-25
  - starting 9-17
  - system configuration options 9-11 to 9-13
- CIMS Windows Process data collector
  - about 3-2
  - installing 3-9 to 3-10
  - job file process example
    - on central CIMS Data Collectors Server 3-19
    - on other server 3-23
  - kernel and user mode, about 3-17
  - log files
    - enabling the creation of 3-11 to 3-13
    - format of 3-13 to 3-16
    - identifiers and resources collected from 3-18
  - setting up 3-19 to 3-24
  - starting 3-13
  - system configuration options 3-4 to 3-8
- CIMSAcct program
  - about 2-9
  - example of use in job file 2-35
  - parameters for 2-65 to 2-69
- CIMSAggregation.dll *See* CIMS Aggregation Engine
- CIMSBill program
  - about 2-10
  - example of use in job file 2-35
  - parameters for 2-71 to 2-75
- CIMSJob.xsd schema, about 2-5
- CIMSLIB.wsf script, about 2-14
- CIMSPrat program
  - about 2-10
  - example of use in job file 2-33
  - files used by
    - parameters file 2-17
    - proration table 2-16
  - parameters for 2-64
- CIMSSort program
  - about 2-10
  - example of use in job file 2-35
  - parameters for 2-70
- CIMSUtills.wsc script, about 2-14
- CIMSWinDisk.xml file, structure of 7-5 to 7-8
- CIMSWinEventLog.xml file, structure of 9-5 to 9-7
- CIMSWinPrint.wsf script, parameters for 9-21 to 9-22
- CIMSWinProcess.wsf script, parameters for 3-20 to 3-21
- Citrix data collector
  - about 3-28
  - identifiers and resources collected by 3-28
  - job file process example 3-29
  - setting up 3-29 to 3-30
- Citrix.wsf script, parameters for 3-30

- Cleanup program
  - about 2-11
  - example of use in job file 2-30, 2-43, 2-44
  - parameters for 2-85 to 2-86
- COM objects
  - CIMSAggregation.dll *See* CIMS Aggregation Engine
- Console program type, parameters for 2-87 to 2-88
- Conversion Builder *See* CIMS Conversion Builder
- conversion definition
  - about 14-2
  - creating 14-3 to 14-23
    - example 14-28 to 14-36
  - opening in CIMS Conversion Builder 14-24
  - saving in CIMS Conversion Builder 14-24
- Conversion Engine *See* CIMS Conversion Engine
- conversion scripts
  - about 2-6
  - parameters for
    - Apache.wsf 6-34
    - CIMSWinPrint.wsf 9-21 to 9-22
    - CIMSWinProcess.wsf 3-20 to 3-21
    - Citrix.wsf 3-30
    - DBSpace.wsf 4-35 to 4-36
    - MSExchange2000.wsf 5-13 to 5-14
    - MSExchange2003.wsf 5-13
    - MSExchange55.wsf 5-13 to 5-14
    - MSExchangeMbx.wsf 5-21 to 5-23
    - MSIIS.wsf 6-9 to 6-10
    - MSISA.wsf 6-18
    - MSProxy.wsf 6-26
    - MSSQL2000.wsf 4-16 to 4-17
    - Netflow.wsf 8-4
    - sendmail.wsf 6-31
    - SQUID.wsf 6-28 to 6-29
    - Transactions.wsf 10-5
    - Universal.wsf 14-27
    - VMware.wsf 3-33
  - standard parameters for 2-7
- D**
- database
  - collectors 4-3 to 4-36
  - defining for the data collection process 2-49
  - program for loading 2-10
- DB2 data collector (Windows)
  - about 4-26
  - job file process example 4-31
  - logging, enabling 4-27 to 4-28
  - resources collected by 4-29 to 4-30
  - running 4-32
  - setting up 4-18 to 4-20, 4-26
- DBLoad program
  - about 2-10
  - example of use in job file 2-30, 2-43, 2-44
  - parameters for 2-76 to 2-78
- DBSpace data collector
  - about 4-33
  - identifiers and resources collected by 4-33
  - job file process example 4-34
  - setting up 4-33 to 4-36
- DBSpace.wsf script, parameters for 4-35 to 4-36
- Default element
  - about 2-90
  - attributes for 2-90
  - example 2-91
- Defaults element, about 2-90
- definition file *See* conversion definition
- E**
- Event Viewer, setting options for CIMS Windows
  - Event Log data collector 9-9
- Evolve data collector 13-1
- Exchange Server 2000 data collector
  - about 5-6
  - job file process example 5-12
  - log files
    - enabling the creation of 5-6
    - format of 5-7 to 5-8
    - identifiers and resources collected from 5-9
  - setting up 5-12 to 5-14
- Exchange Server 2003 data collector
  - about 5-6
  - job file process example 5-12
  - log files
    - enabling the creation of 5-6
    - format of 5-7 to 5-8
    - identifiers and resources collected from 5-9
  - setting up 5-12 to 5-14

Exchange Server 5.5 data collector  
about 5-2  
job file process example 5-12  
log files  
enabling the creation of 5-2 to 5-3  
format of 5-3 to 5-4  
identifiers and resources collected from 5-5  
setting up 5-12 to 5-14  
Exchange Server Mailbox data collector  
about 5-15  
identifiers and resources collected by 5-17 to 5-19  
job file process example 5-20  
security permissions required for 5-15  
setting up 5-20 to 5-23  
system requirements 5-15

## F

feed folder  
contents of 2-13  
creating 2-13  
FileTransfer program  
about 2-11  
example of use in job file 2-30, 2-31  
parameters for 2-80 to 2-85  
ftp file transfer  
attributes for 2-80 to 2-85  
example in job file 2-30, 2-31

## G

GenerateExternalFile element  
about 2-59  
attributes for 2-59 to 2-60

## I

IIS data collector  
about 6-3  
job file process example 6-8  
log files  
enabling the creation of 6-3  
format of 6-4 to 6-5  
identifiers and resources collected from 6-6 to 6-7  
setting up 6-8 to 6-10  
installing  
CIMS Data Collectors 2-2  
CIMS Windows Print data collector 9-14 to 9-15

CIMS Windows Process data collector 3-9 to 3-10  
CIMSSp\_SQLServer2000Trace stored procedure 4-4  
ISA Server data collector  
about 6-11  
job file process example 6-17  
log files  
enabling the creation of 6-11  
format of 6-12 to 6-15  
identifiers and resources collected from 6-16  
setting up 6-17 to 6-18

## J

Job element  
about 2-48  
attributes for 2-48 to 2-51  
job files  
about 2-5  
creating 2-25 to 2-91  
Default element  
about 2-90  
attributes for 2-90  
example 2-91  
Defaults element, about 2-90  
GenerateExternalFile element  
about 2-59  
attributes for 2-59 to 2-60  
Job element  
about 2-48  
attributes for 2-48 to 2-51  
Jobs element  
about 2-46  
attributes for 2-46 to 2-47  
Nightly.xml and Monthly.xml sample files 2-5  
Parameter element  
about 2-60  
CIMSAct attributes 2-65 to 2-69  
CIMSBill attributes 2-71 to 2-75  
CIMSSort attributes 2-70  
Cleanup attributes 2-85 to 2-86  
Console program type attributes 2-87 to 2-88  
DBLoad attributes 2-76 to 2-78  
FileTransfer attributes 2-80 to 2-85  
Scan attributes 2-62 to 2-63  
WaitFile attributes 2-78 to 2-80  
Parameters element, about 2-60

- Process element
    - about 2-52
    - attributes for 2-52 to 2-54
  - schema, about 2-5
  - Step element
    - about 2-55
    - attributes for 2-55 to 2-58
  - Steps element
    - about 2-55
    - attribute for 2-55
    - structure of 2-46 to 2-91
    - testing 2-8, 2-94
  - job log files
    - about 2-8
    - defining the type and content of 2-8
    - e-mail distribution of 2-8
    - return codes in 2-9
  - Jobs element
    - about 2-46
    - attributes for 2-46 to 2-47
- K**
- kernel and user mode, about 3-17
- L**
- log dates
    - about 2-3 to 2-4
    - passing
      - default parameter 2-4
      - from the command line 2-4, 2-94
      - from the job file 2-4, 2-26, 2-90
  - Lotus Notes data collector 5-24
- M**
- Mainframe data collector
    - about 11-2
    - job file process example
      - for CIMS Ident, Detail, and Summary file collection 11-5
      - for CSR+ file collection 11-3
    - setting up
      - for CIMS Ident, Detail, and Summary file collection 11-5
      - for CSR or CSR+ file collection 11-3 to 11-4
  - Monthly.xml file, about 2-5
  - MS 2-25
  - MSEExchange2000.wsf script, parameters for 5-13 to 5-14
  - MSEExchange2003.wsf script, parameters for 5-13
  - MSEExchange55.wsf script, parameters for 5-13 to 5-14
  - MSEExchangeMbx.wsf script, parameters for 5-21 to 5-23
  - MSIIS.wsf script, parameters for 6-9 to 6-10
  - MSISA.wsf script, parameters for 6-18
  - MSProxy.wsf script, parameters for 6-26
  - MSSQL2000.wsf script, parameters for 4-16 to 4-17
- N**
- NetBackup data collector
    - about 7-9
    - identifiers and resources collected by 7-10
    - log files
      - format of 7-9
      - setting up 7-11 to 7-12
  - NetFlow data collector
    - about 8-2
    - identifiers and resources collected by 8-2
    - job file process example 8-3
    - setting up 8-3 to 8-4
  - Netflow.wsf script, parameters for 8-4
  - Netscape Proxy Server data collector 6-35
  - Nightly.xml job file, about 2-5
  - Novell NetWare data collector 8-5
- O**
- Oracle data collector (Windows)
    - about 4-18
    - job file process example 4-24
    - logging, enabling 4-21 to 4-22
    - resources collected by 4-23 to 4-24
    - running 4-25
    - setting up 4-18 to 4-21
  - Outlook Web Access data collector 5-24
- P**
- Parameter element
    - about 2-60
    - attributes
      - CIMSACCT program 2-65 to 2-69
      - CIMSBill program 2-71 to 2-75
      - CIMSSort program 2-70
      - Cleanup program 2-85 to 2-86
      - Console program type 2-87 to 2-88
      - DBLoad program 2-76 to 2-78

- FileTransfer program 2-80 to 2-85
- Scan program 2-62 to 2-63
- WaitFile program 2-78 to 2-80
- Parameters element, about 2-60
- printer data collectors *See* CIMS Windows Event
  - Log data collector and CIMS Windows
  - Print data collector
- process definition folders
  - about 2-12
  - contents of 2-13
  - creating 2-12
- Process element
  - about 2-52
  - attributes for 2-52 to 2-54
- Processes folder
  - creating 2-12
  - path to, defining 2-12
- processing data
  - suggested frequency 2-92
- processing date
  - permissions (folder) required for 2-92
- processing programs, descriptions of 2-9 to 2-11
- proration
  - about 2-15
  - example 2-21 to 2-24
  - parameters file for
    - creating 2-17 to 2-21
    - example 2-22 to 2-23
  - table
    - creating 2-16
    - example 2-21
- Proxy Server data collector
  - about 6-19
  - job file process example 6-25
  - log files
    - enabling the creation of 6-19
    - format of 6-20 to 6-24
    - identifiers and resources collected from 6-24
  - setting up 6-25 to 6-26

**R**

- Reporting Services data collector
  - about 4-37
  - identifiers and resources collected by 4-37
  - job file process example 4-38
  - setting up 4-38 to 4-40
- return codes, descriptions of 2-9
- running CIMS Data Collectors 2-93 to 2-95

**S**

- SAP data collector 13-1
- Scan program
  - about 2-9
  - example of use in job file 2-30, 2-42
  - parameters for 2-62 to 2-63
  - Smart Scan feature
    - enabling 2-42 to 2-45
    - example of use in job file 2-42 to 2-45
- schema for job files, about 2-5
- sendmail data collector
  - about 6-30
  - identifiers and resources collected by 6-30
  - job file process example 6-30
  - setting up 6-30 to 6-32
- sendmail.wsf script, parameters for 6-31
- Shell.wsc script, about 2-14
- Shiva data collector 13-1
- SingleProcessStep program
  - about 2-10
  - example of use in job file 2-30, 2-43, 2-44
- Smart Scan
  - enabling 2-42 to 2-45
  - example of use in job file 2-42 to 2-45
- SQL Server 2000 data collector
  - about 4-3
  - job file process examples 4-9 to 4-15
  - setting up 4-9 to 4-17
  - stored procedures
    - CIMSSp\_SQLServer2000Trace 4-3, 4-4
    - SQLServer2000Trace 4-6
  - trace files
    - enabling creation of 4-4 to 4-6
    - format of 4-6 to 4-7
    - identifiers and resources collected from 4-8
- SQL Server Reporting Services data collector *See* Reporting Services data collector
- SQUID data collector
  - about 6-27
  - identifiers and resources collected by 6-27
  - job file process example 6-27
  - setting up 6-27 to 6-29
- SQUID.wsf script, parameters for 6-28 to 6-29
- Step element
  - about 2-55
  - attributes for 2-55 to 2-58

Steps element  
 about 2-55  
 attribute for 2-55  
 stored procedures  
 CIMSSp\_SQLServer2000Trace  
 about 4-3  
 installing 4-4  
 running 4-5  
 SQLServer2000Trace, modifying 4-6  
 Sybase data collector 4-33  
 system  
 architecture, described 2-3 to 2-14  
 specifications 2-2

## T

Transactions data collector  
 about 10-2  
 CIMSTransactions table  
 format of 10-2 to 10-3  
 identifiers and resources collected from 10-4  
 job file process example 10-4  
 setting up 10-4 to 10-6  
 Transactions.wsf script, parameters for 10-5  
 transferring files  
 job file example 2-30, 2-31  
 program for 2-11

## U

Universal data collector  
 about 14-2  
 creating a conversion definition for 14-3 to 14-23  
 example 14-28 to 14-36  
 data conversion process 14-2  
 job file process example 14-25  
 rate codes for  
 adding to CIMSRate table 14-25  
 defining in the conversion definition 14-18  
 setting up 14-25 to 14-27  
 Universal.wsf script, parameters for 14-27  
 UNIX data collector  
 about 12-2  
 job file process example 12-2  
 setting up 12-2 to 12-3

## V

variable values used for data collection 2-61  
 Veritas data collector 7-12

VMware data collector  
 about 3-31  
 identifiers and resources collected by 3-31  
 job file process example 3-32  
 setting up 3-32 to 3-33  
 VMware.wsf script, parameters for 3-33

## W

WaitFile program  
 about 2-11  
 example of use in job file 2-32  
 parameters for 2-78 to 2-80  
 Windows disk storage data collector *See* CIMS  
 Windows Disk data collector and  
 DiskDir data collector  
 Windows operating system data collector *See*  
 CIMS Windows Processes data collector  
 WSRM data collector  
 about 3-25  
 identifiers and resources collected by 3-25  
 job file process example 3-26  
 setting up 3-26 to 3-27

