IBM Tivoli Usage and Accounting Manager Data Collectors
for Microsoft Windows

User’s Guide

Version 5.1
Note • Before using this information and the product it supports, read the information in Appendix C, Notices.
# Table of Contents

## Preface

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who Should Read This Guide</td>
<td>vii</td>
</tr>
<tr>
<td>What This Guide Contains</td>
<td>viii</td>
</tr>
<tr>
<td>Related Publications</td>
<td>viii</td>
</tr>
<tr>
<td>Contacting Customer Support</td>
<td>ix</td>
</tr>
<tr>
<td>Conventions Used in This Guide</td>
<td>ix</td>
</tr>
<tr>
<td>Terminology Used in this Guide</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1 • About ITUAM Data Collectors

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Are ITUAM Data Collectors?</td>
<td>1-2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Specifications</td>
<td>2-3</td>
</tr>
<tr>
<td>Installing ITUAM Data Collectors</td>
<td>2-3</td>
</tr>
<tr>
<td>ITUAM Data Collectors Architecture</td>
<td>2-4</td>
</tr>
<tr>
<td>ITUAM Job Runner Program</td>
<td>2-4</td>
</tr>
<tr>
<td>Job Files (JobFiles Folder)</td>
<td>2-5</td>
</tr>
<tr>
<td>Job File XML Schema</td>
<td>2-6</td>
</tr>
<tr>
<td>Collection Files (Collectors Folder)</td>
<td>2-7</td>
</tr>
<tr>
<td>Job Log Files (LogFiles Folder)</td>
<td>2-9</td>
</tr>
<tr>
<td>ITUAM Processing Programs (Process Engine\JobLibrary Folder)</td>
<td>2-10</td>
</tr>
<tr>
<td>Process Definitions (Processes Folder)</td>
<td>2-13</td>
</tr>
<tr>
<td>Scripts (Scripts Folder)</td>
<td>2-15</td>
</tr>
<tr>
<td>Setting Up Proration Files (Optional)</td>
<td>2-16</td>
</tr>
<tr>
<td>Creating a Proration Table</td>
<td>2-17</td>
</tr>
<tr>
<td>Creating the CIMSPrat Parameters File</td>
<td>2-18</td>
</tr>
<tr>
<td>Proration Example</td>
<td>2-23</td>
</tr>
</tbody>
</table>
Table of Contents

Creating Job Files .................................................................................................................. 2-27
Using ITUAM Date Keywords in the Job File ......................................................................... 2-27
Using Log Dates in the Job File ............................................................................................ 2-28
Using the SampleNightly_xSeries.xml Job File .................................................................... 2-29
Using the SampleNightly_TDS.xml Job File .......................................................................... 2-35
Example of Reprocessing With Smart Scan Enabled ............................................................. 2-40
Job File Structure ................................................................................................................ 2-44

Using ITUAM Integrator ....................................................................................................... 2-90
Input Element ....................................................................................................................... 2-91
Stage Elements .................................................................................................................. 2-91

Running ITUAM Data Collectors .......................................................................................... 2-119
Data Processing Frequency .................................................................................................. 2-119
Required Folder Permissions for Data Processing ............................................................... 2-119
Running ITUAM Job Runner ............................................................................................... 2-120

3 • Operating System Data Collectors

Windows Process Data Collector ............................................................................................. 3-2
Creating a Log On User Account for the Windows Process Collector Service (Optional) ........ 3-2
System Configuration Options for the Windows Process Collector ........................................ 3-4
Installing the Windows Process Collector ............................................................................ 3-9
Windows Process Collector Log File Format ....................................................................... 3-21
Identifiers and Resources Collected From the Windows Process Collector Log File ............... 3-25
Setting Up the Windows Process Collector ........................................................................... 3-26
Running the Windows Process Collector .............................................................................. 3-32

VMware Data Collector ......................................................................................................... 3-33
Identifiers and Resources Collected by the VMware Collector .............................................. 3-33
Setting Up the VMware Collector ....................................................................................... 3-34
Running the VMware Collector ......................................................................................... 3-36

4 • Storage Data Collectors

Windows Disk Data Collector ................................................................................................. 4-2
Identifiers and Resources Collected by the Windows Disk Collector ..................................... 4-2
Setting Up the Windows Disk Collector ................................................................................ 4-3
Running the Windows Disk Collector .................................................................................. 4-9
5 • Tivoli Decision Support for z/OS (TDSz) Collector

About TDSz Data Collection .................................................................................... 5-2
Installing and Configuring DB2 Connect ................................................................. 5-2
Setting Up Conversion Definitions ....................................................................... 5-4
About ITUAM Conversion Engine and Conversion Builder .............................. 5-5
Working With Conversion Definitions ................................................................. 5-5
Running ITUAM Conversion Engine from ITUAM Conversion Builder .......... 5-14
Creating a Job File for the TDSz Collector ......................................................... 5-15
Collection Step Parameters ................................................................................. 5-17
Running the TDSz Collector Job File .................................................................. 5-18

6 • Processing Data from ITUAM Data Collectors for UNIX/Linux

About UNIX/Linux Data Collection ................................................................... 6-2
Setting Up Collection of UNIX/Linux CSR Files .............................................. 6-2
Running the UNIX/Linux CSR File Collection Process .................................... 6-4

A • ITUAM Aggregation Engine API

About ITUAM 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
ExceptionFile Interface Method ........................................................................... A-18
### B • Conversion Definition File Description

- **Overview** ................................................................. B-2
- **Conversion Definition Components** ........................................ B-3
  - Input Parameters ......................................................... B-3
  - Output Parameters ....................................................... B-7
  - Field Definition Parameters ........................................... B-9
  - Identifier Parameters ................................................... B-15
  - Resource Parameters .................................................... B-16
  - Date/Time Parameters ................................................... B-19
  - Shift Parameters ....................................................... B-21

### C • Notices

- **Trademarks** .............................................................. C-3

### Glossary

### Index
Preface

This guide provides a comprehensive set of procedures and references for installing, configuring, and managing the components that compose the IBM® Tivoli® Usage and Accounting Manager data collection system.

For the purposes of world-wide marketing, IBM Tivoli Usage and Accounting Manager is sometimes referred to as IBM Usage and Accounting Manager or ITUAM. Users of this product may see these names in presentations and product collateral. For the purposes of this guide, the product is referred to as ITUAM unless referring to another document or entity that contains the full name.

Who Should Read This Guide

This guide is for system administrators responsible for installing, setting up, and running ITUAM 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)

ITUAM Data Collectors are components of ITUAM. This guide assumes that the ITUAM system is installed and configured as described in the IBM Tivoli Usage and Accounting Manager Administrator’s Guide and that users are familiar with concepts associated with ITUAM, including the layout and use of CSR files.
What This Guide Contains

The following table describes the chapters in this guide. You should begin with Chapter 2, Installing ITUAM Data Collectors and Setting Up the System and then continue to the collector-specific information provided in Chapter 3 through Chapter 6.

<table>
<thead>
<tr>
<th>Ch. No.</th>
<th>Chapter Name</th>
<th>Content Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>About ITUAM Data Collectors</td>
<td>Provides an introduction to ITUAM Data Collectors.</td>
</tr>
<tr>
<td>2</td>
<td>Installing ITUAM Data Collectors and Setting Up the System</td>
<td>Provides steps for installing and setting up ITUAM Data Collectors and an overview of the system architecture.</td>
</tr>
<tr>
<td>3</td>
<td>Operating System Data Collectors</td>
<td>Provides logging and setup procedures for ITUAM Data Collectors for operating systems.</td>
</tr>
<tr>
<td>4</td>
<td>Storage Data Collectors</td>
<td>Provides setup procedures for ITUAM Data Collectors for storage systems.</td>
</tr>
<tr>
<td>5</td>
<td>Tivoli Decision Support for z/OS (TDSz) Collector</td>
<td>Provides setup procedures for the ITUAM Data Collector for TDSz.</td>
</tr>
<tr>
<td>6</td>
<td>Processing Data from ITUAM Data Collectors for UNIX/Linux</td>
<td>Provides setup procedures for processing CSR files created by ITUAM Data Collectors for UNIX and Linux.</td>
</tr>
<tr>
<td>A</td>
<td>ITUAM Aggregation Engine API</td>
<td>Describes the ITUAM Aggregation Engine API.</td>
</tr>
<tr>
<td>B</td>
<td>Conversion Definition File Description</td>
<td>Provides a description of the conversion definition file used by some collectors to convert data.</td>
</tr>
<tr>
<td>C</td>
<td>Notices</td>
<td>Provides licensing, copyright, and trademark information.</td>
</tr>
</tbody>
</table>

Related Publications

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

- IBM Tivoli Usage and Accounting Manager Administrator's Guide
- IBM Tivoli Usage and Accounting Manager Data Collectors for UNIX and Linux User's Guide
Contacting Customer Support

Before contacting IBM Tivoli Software Support with a problem, refer to the IBM Tivoli Software Support site by clicking the Tivoli link at the following Web site:

http://www.ibm.com/software/support/

If you need additional help, contact software support by using the methods described in the IBM Software Support Guide at the following Web site:

http://techsupport.services.ibm.com/guides/handbook.html

The IBM Software Support Guide provides the following information:

- Registration and eligibility requirements for receiving support.
- Telephone numbers, depending on the country in which you are located.
- A list of information you should gather before contacting technical support.

Conventions Used in This Guide

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

<table>
<thead>
<tr>
<th>Symbol or Type Style</th>
<th>Represents</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate color</td>
<td>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</td>
<td>.....see Data Migration.</td>
</tr>
<tr>
<td>Italic</td>
<td>words that are emphasized</td>
<td>...the entry after the current entry...</td>
</tr>
<tr>
<td></td>
<td>a new term</td>
<td>...by identifier values.</td>
</tr>
<tr>
<td></td>
<td>the titles of other manuals</td>
<td>IBM Tivoli Usage and Accounting Manager Administrator’s Guide</td>
</tr>
<tr>
<td></td>
<td>variables in file names</td>
<td>CIMSProcessLog-yyyymmdd.txt</td>
</tr>
<tr>
<td>Bold</td>
<td>names of interface items such as tabs, boxes, buttons, lists, and check boxes.</td>
<td>Select the Use Local Time check box Enter the path in the Log File Path box</td>
</tr>
<tr>
<td>Monospace</td>
<td>directories, file names, command names, computer code, computer screen text, system responses, command line commands, what the user types</td>
<td>Processes folder MSSQL2000.wsf script</td>
</tr>
</tbody>
</table>
### Terminology Used in this Guide

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

<table>
<thead>
<tr>
<th>Symbol or Type Style</th>
<th>Represents</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&gt;</td>
<td>the name of a key on the keyboard</td>
<td>Press &lt;Enter&gt;</td>
</tr>
<tr>
<td>‣</td>
<td>navigating a menu or a folder</td>
<td>File ‣ Import ‣ Object</td>
</tr>
</tbody>
</table>
About ITUAM Data Collectors

What Are ITUAM Data Collectors? ................................................................. 1-2
About ITUAM Data Collectors

What Are ITUAM Data Collectors?

ITUAM 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 ITUAM. This output file is called a CSR file.

ITUAM 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.

This guide describes the data collectors that run on the Windows operating system to collect usage data from Windows and Windows-compatible applications (including older Windows operating systems such as Windows NT®) and non-Windows applications and operating systems.

All references to ITUAM Data Collectors in the following chapters refer to the collectors that run on the Windows system.
Installing ITUAM Data Collectors and Setting Up the System

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

System Specifications ................................................................. 2-3
Installing ITUAM Data Collectors .................................................. 2-3
ITUAM Data Collectors Architecture .............................................. 2-4
ITUAM Job Runner Program ....................................................... 2-4
Job Files (JobFiles Folder) .......................................................... 2-5
Job File XML Schema ............................................................... 2-6
Collection Files (Collectors Folder) ............................................... 2-7
Job Log Files (LogFiles Folder) .................................................... 2-9
ITUAM Processing Programs (Process Engine\JobLibrary Folder) ........ 2-10
Process Definitions (Processes Folder) .......................................... 2-13
Scripts (Scripts Folder) .............................................................. 2-15
Setting Up Proration Files (Optional) ........................................... 2-16
Creating a Proration Table ........................................................ 2-17
Creating the CIMSPrat Parameters File ........................................ 2-18
Proration Example ................................................................. 2-23
Creating Job Files ................................................................. 2-27
Using ITUAM Date Keywords in the Job File ............................... 2-27
Using Log Dates in the Job File .................................................... 2-28
Using the SampleNightly_ xSeries.xml Job File ............................... 2-29
Using the SampleNightly_TDS.xml Job File ................................ 2-35
Example of Reprocessing With Smart Scan Enabled ....................... 2-40
Job File Structure ................................................................. 2-44
Installing ITUAM Data Collectors and Setting Up the System

Using ITUAM Integrator ................................................................. 2-90
Input Element ............................................................................. 2-91
Stage Elements ....................................................................... 2-91

Running ITUAM Data Collectors .................................................... 2-119
Data Processing Frequency ......................................................... 2-119
Required Folder Permissions for Data Processing ......................... 2-119
Running ITUAM Job Runner ....................................................... 2-120
System Specifications

The following are the system specifications for running ITUAM Data Collectors. Note that you can use ITUAM Data Collectors to process usage metering data.

- Microsoft Windows 2000 Server or Windows Server 2003 with the latest service pack.
- Microsoft .NET Framework 1.1. The ITUAM setup program (which installs ITUAM 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 ITUAM Data Collectors

The ITUAM setup program includes ITUAM Data Collectors. When you install ITUAM, you can choose to install all or certain ITUAM Data Collectors (refer to the IBM Tivoli Usage and Accounting Manager Administrator’s Guide for the installation procedures).

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

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 Windows Process collector, which produces log files containing operating system data, on another computer.

  ITUAM includes a simple setup program for installing the Windows Process collector on other computers. These setup programs install the executable and administrative programs and conversion script for the collector and ITUAM Aggregation Engine. For installation and setup procedures for this collector, see Chapter 3, Operating System 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 ITUAM Aggregation Engine on the computer. Contact IBM for assistance.
ITUAM Data Collectors Architecture

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

The components are grouped by folder in C:\Program Files\IBM\ITUAM (if you installed ITUAM 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 ITUAM Data Collectors usually do not require modification. However, if you modify any file that is provided in the IBM\ITUAM folder, it is very important that you rename the file. Otherwise, the file will be overwritten when you upgrade to a new version of ITUAM Data Collectors.

ITUAM Job Runner Program

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

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

You can run ITUAM Job Runner directly from the command prompt or you can use Windows Task Scheduler to schedule the program to run automatically (see Running ITUAM Job Runner on page 2-120).

Specifying Log Dates for Collection

ITUAM 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.)
- **PREWEEK** (Collects data produced in the previous week [Sun–Sat].)
- **PREMONTH** (Collects data produced in the previous month.)
- **CURWEEK** (Collects data produced in the current week [Sun–Sat].)
- **CURMONTH** (Collects data produced in the current month.)
- date in yyyymmd format (Collects data produced on a specified date.)
- date range in yyyymmd yyyymmd 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 ITUAM 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, ITUAM 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 ITUAM Job Runner (see Running ITUAM Job Runner on page 2-120).

When you enter a LogDate parameter that includes a date range, such as CURMON, ITUAM 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 if you are running a snapshot collector (for example, the Windows Disk collector). Snapshot collectors collect data that is current as of the date and time that the collectors are run by ITUAM 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.

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 ITUAM components required to process the data and load it into an ITUAM 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.)

ITUAM includes a sample job file that you can modify for your organization, SampleNightly_xSeries.xml or SampleNightly_TDS.xml. The sample job file that you should use depends on the ITUAM package that you are using (ITUAM [...xSeries.xml] or ITUAM for z/OS [...TDS.xml]).

If you modify any sample job file, you need to rename it. Otherwise, the file will be overwritten when you install a new version of ITUAM Data Collectors. If you installed ITUAM Data Collectors in the default location, the sample job files are in C:\Program Files\IBM\ITUAM\JobFiles.
Note • The JobFiles folder also contains sample job files other than SampleNightly_xSeries.xml and SampleNightly_TDS.xml. These files are described as applicable in this and other ITUAM guides.

The sample job files are intended to be run on a nightly basis to run one or multiple data collectors. However, you can schedule ITUAM Job Runner to run job files on any schedule.

For a description of the job file structure, see Job File Structure on page 2-44.

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\IBM\ITUAM\Process Engine\JobLibrary (if you installed ITUAM 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 ITUAM 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 ITUAM 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 ITUAM Aggregation Engine (if applicable). ITUAM 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, ITUAM 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 ITUAM Processing Engine must process and improves processing time.

  For more information about the ITUAM Aggregation Engine, see Appendix A, ITUAM 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.)

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

For many collectors, the resources defined in the conversion script are preloaded as rate codes in CIMSRate table. You can then use ITUAM 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 preloaded in the CIMSRate table and must be added as described in the IBM Tivoli Usage and Accounting Manager 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogDate</td>
<td>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 Specifying Log Dates for Collection on page 2-4.</td>
</tr>
<tr>
<td>RetentionFlag</td>
<td>This parameter is for future use.</td>
</tr>
<tr>
<td>Feed</td>
<td>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 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. A subfolder with the same name as the feed 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 from a usage metering file (see Feed Subfolder on page 2-14). This is the CSR file that is processed by the Scan program. This parameter is included as an identifier in the CSR file. The identifier name is Feed and the identifier value is the server name.</td>
</tr>
<tr>
<td>OutputFolder</td>
<td>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program (see page 2-10). This parameter is defined by the Process id for the collector. For example, Process id=&quot;VMware&quot; specifies that the output folder is the VMware process definition folder. For more information about the process definition folder, see page 2-13.</td>
</tr>
</tbody>
</table>

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 before scheduling the job to run in batch (see To run ITUAM Job Runner from the command prompt: on page 2-121).

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).

**Note** • If you want to use the optional ITUAM Web Console application, the job log files must be in XML format. For more information about this application, refer to the IBM Tivoli Usage and Accounting Manager Administrator's Guide.

You can also send log data to the ITUAM database using the attribute joblogWriteToDB="true".

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.

ITUAM 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).
- CIMSAcct, 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 2-40.

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

For the parameters used by the CIMSAcct program, see page 2-65.
CIMSSort
The CIMSSort program sorts the CIMSAcct 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 CIMSAcct Detail file from CIMSSort and performs shift processing, CPU normalization, and include/exclude processing and creates the CIMSBill Detail and Summary files. These files contain the billing information used to generate invoices and reports.

Note • Although you can perform proration using CIMSBill, the CIMSPrat program provides more options for proration and enables you to prorate resources multiple times.

For the parameters used by the CIMSBill program, see page 2-71.

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

DBLoad
The DBLoad program loads the output files from CIMSAcct and CIMSBill into the ITUAM 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 2-16.
**WaitFile**
The WaitFile program directs ITUAM Job Runner to wait for one or more files before continuing processing.

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 ITUAM server.

For the parameters used by the FileTransfer program, see page 2-79.

**Cleanup**
The Cleanup program deletes files with file names containing the date in yyyymmdd 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 2-84.
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 2-13).

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 ITUAM Administrator (refer to the IBM Tivoli Usage and Accounting Manager 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, ITUAM Job Runner can create a folder using the process ID defined in the job file as the folder name.

About the Processes Folder

The folder Sample Processes is shipped with ITUAM Data Collectors. When ITUAM 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. It is recommended 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 ITUAM Administrator. The path is stored in the CIMSGlobalOptions table and is used as the default path for the process definition folders. To set the Processes folder path, refer to the IBM Tivoli Usage and Accounting Manager Administrator’s Guide.

Each time that you upgrade to a new release of ITUAM 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 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 4.

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.

The Scan program processes and concatenates the CSR files in the feed subfolders as described in Scan on page 2-10. 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 Windows Disk collector includes an XML file (CIMSWindDisk.xml) that provides parameters used by this collector. For more information, see Chapter 4.

Additional Processing Files

Each process definition folder contains additional processing files that are used internally by ITUAM Data Collectors.
Scripts (Scripts Folder)

The ITUAM 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 ITUAM 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.

Although you can perform proration using CIMSBill, the CIMSPrat program provides more options for proration and enables you to prorate resources multiple times.

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.

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 2-17.

- 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 2-18.

  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 2-23.

CIMSPrat is run as a step in a job file. An example of the use of CIMSPrat in a job file is provided in the sample job file SampleNightly.xml (see the UnixFS process).

The SampleNightly.xml file is not intended to replace the sample job files described in Job Files (JobFiles Folder) on page 2-5. The purpose of the SampleNightly.xml file is to show different configuration options that you might want to implement for the data collection process such as collecting from multiple servers, prorating resources, and running the processing programs (CIMSAcct, CIMSSort, and CIMSBill) in separate steps rather than as one step.
Creating a Proration Table

The ITUAM 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 ITUAM 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 2-19). 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 2-23.
Creating the CIMSPrat Parameters File

The ITUAM 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 is not overwritten when you upgrade to a new version of ITUAM Data Collectors.

For an example of a modified CIMSPrat parameters file used in the proration process, see Proration Example on page 2-23.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputFile and OutputFile</td>
<td>Required</td>
<td>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.</td>
</tr>
<tr>
<td>ProrateFile</td>
<td>Required</td>
<td>The full path for the proration table. The proration table can be in any folder.</td>
</tr>
<tr>
<td>PrintFile</td>
<td>Required</td>
<td>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.</td>
</tr>
<tr>
<td>ExceptionFile</td>
<td>Required</td>
<td>The full path for the output exception file. This file contains records that do not include an identifier name that matches the IdentifierName 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 ExceptionProcessing attribute in the parameters file set to &quot;TRUE&quot;. If you have ExceptionProcessing set to &quot;False&quot;, unmatched records are included in the prorated CSR or CSR+ file.</td>
</tr>
</tbody>
</table>

Table 2-2 • CIMSPrat.xml Parameters
Setting Up Proration Files (Optional)

Audit
Optional
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 page 2-25).

- The input identifier name prefixed by Orig_ and the original identifier value. (This is not applicable if the DiscardIdentifier attribute in the parameter file is set to "TRUE" and/or NewIdentifier attribute is set to a new value).

- An additional rate code, ProratePct, that provides the proration percentage value.

- The input rate code prefixed by Orig_ and the original resource value.

Valid values are:
- "TRUE" (audit is enabled)
- "FALSE" (audit is not enabled)

The default is "TRUE".

IdentifierName
Required
The name of the identifier field that you want to use to select CSR or CSR+ records for proration.

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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| Audit            | Optional             | 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 page 2-25).

- The input identifier name prefixed by Orig_ and the original identifier value. (This is not applicable if the DiscardIdentifier attribute in the parameter file is set to "TRUE" and/or NewIdentifier attribute is set to a new value).

- An additional rate code, ProratePct, that provides the proration percentage value.

- The input rate code prefixed by Orig_ and the original resource value.

Valid values are:
- "TRUE" (audit is enabled)
- "FALSE" (audit is not enabled)

The default is "TRUE".

IdentifierName
Required
The name of the identifier field that you want to use to select CSR or CSR+ records for proration.

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.

Table 2-2 • CIMPrat.xml Parameters (Continued)
### Setting Up Proration Files (Optional)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowNon100Totals</td>
<td>Optional</td>
<td>Specifies whether the total prorate percentages for an identifier name must equal 100 percent. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;TRUE&quot; (percentages are not required to equal 100 percent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;FALSE&quot; (percentages must equal 100 percent)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;TRUE&quot;.</td>
</tr>
<tr>
<td>IdentifierStart</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is 1.</td>
</tr>
<tr>
<td>IdentifierLength</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value begins at the position specified by the IdentifierStart attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is 8.</td>
</tr>
<tr>
<td>MaximumRecords</td>
<td>Option</td>
<td>The maximum number of records that you want to process in the CSR or CSR+ files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is all records in the file.</td>
</tr>
<tr>
<td>PrintLines</td>
<td>Optional</td>
<td>The number of lines per page in the CIMSPrat processing report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is 60.</td>
</tr>
<tr>
<td>ExceptionProcessing</td>
<td>Optional</td>
<td>Specifies whether the exception file should be created. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;TRUE&quot; (the exception file is created)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;FALSE&quot; (the exception file is not created)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;FALSE&quot;.</td>
</tr>
</tbody>
</table>

*Table 2-2 • CIMSPrat.xml Parameters (Continued)*
### Setting Up Proration Files (Optional)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewIdentifier</td>
<td>Optional</td>
<td>Specifies that a new identifier name will appear for the output identifier value in the prorated records. The default is to use the original identifier name. For an example of the use of this attribute, see Proration Example on page 2-23.</td>
</tr>
</tbody>
</table>
| DiscardIdentifier | Optional             | Specifies that the original identifier name and value will not appear in the prorated records. Valid values are:  
  - "TRUE" (the original identifier name and value are discarded)  
  - "FALSE" (the original identifier name and value are retained)  
  The default is "FALSE".  
  This attribute is useful in situations where the original identifier value is not intended for public view (for example, a social security number).  
  This attribute is intended to be used with the NewIdentifier attribute. If you do not include the NewIdentifier attribute, neither the input nor output identifier names and values will appear in the prorated records.  
  For an example of the use of this attribute, see Proration Example on page 2-23. |
| Test            | Optional             | A numeric value used as a test flag. Used by IBM technical support only. |

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

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 page 2-24.

**CatchallIdentifier**
- Optional
  - Specifies the identifier value to be used for records with identifier values that do not have a match in the proration table.
  - If you want to use catchall processing, you must include this parameter, regardless of whether you specify a value or accept the default (`Catchall`).
  - Once you uncomment this parameter, you can leave the remaining catchall parameters commented, and the defaults will be used.

**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>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatchallIdentifier</td>
<td>Optional</td>
<td>Specifies the identifier value to be used for records with identifier values that do not have a match in the proration table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Once you uncomment this parameter, you can leave the remaining catchall parameters commented, and the defaults will be used.</td>
</tr>
<tr>
<td>CatchallPercent</td>
<td>Optional</td>
<td>Specifies the proration percentage to be used for records with identifier values that do not have a match in the proration table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is 100.</td>
</tr>
<tr>
<td>CatchallRate</td>
<td>Optional</td>
<td>Specifies the rate code(s) to be used for records with identifier values that do not have a match in the proration table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is all rate codes.</td>
</tr>
</tbody>
</table>
Proration Example

Your organization uses software program, ABCPrint, that produces usage metering data by printer. This data is collected and provided in 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,20060916,20060916,13:01:50,13:01:50,,1,PrinterName,"LaserJ",2,SUBBYTE,5107,PRNTBYTE,5107
ABCPrint,20060916,20060916,16:15:03,16:15:03,,1,PrinterName,"OptraL",2,SUBBYTE,1913,PRNTBYTE,1913
ABCPrint,20060916,20060916,17:13:33,17:13:33,,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 2-20).

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 2-24).

```
LaserJ,MikeR,25,All
LaserJ,Robert,25,All
LaserJ,Joan,50,All
OptraL,MikeL,25,PRNTBYTE
OptraL,Bill,25,PRNTBYTE
OptraL,Mark,25,PRNTBYTE
OptraL,Tom,25,PRNTBYTE
```
CIMSPPrat Parameters File Example

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

```xml
<?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\IBM\ITUAM\Processes\ABCPrint\CurrentCSR.TXT"
    OutputFile="C:\Program Files\IBM\ITUAM\Processes\ABCPrint\ProRatedCurrentCSR.TXT"
    ProrateFile="C:\Program Files\IBM\ITUAM\Processes\ABCPrint\Prorate.txt"
    PrintFile="C:\Program Files\IBM\ITUAM\Processes\ABCPrint\PRATPRNT.TXT"
    ExceptionFile="C:\Program Files\IBM\ITUAM\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 IBM 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. -->
  <Parameter CatchallIdentifier="Unassigned_Mktg" />
  <Parameter CatchallIdentifier="Unassigned_Sales" />
  <Parameter CatchallIdentifier="CATCH30" />
</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 2-26.

1. ABCPrint,20060916,20060916,13:01:50,13:01:50,,4,User,MikeR,ProratePct,25,Orig_SUBBYTE,5107,Orig_PRNTBYTE,5107,2,SUBBYTE,1276.75,PRNTBYTE,1276.75
2. ABCPrint,20060916,20060916,13:01:50,13:01:50,,4,User,Robert,ProratePct,25,Orig_SUBBYTE,5107,Orig_PRNTBYTE,5107,2,SUBBYTE,1276.75,PRNTBYTE,1276.75
3. ABCPrint,20060916,20060916,13:01:50,13:01:50,,4,User,Joan,ProratePct,50,Orig_SUBBYTE,5107,Orig_PRNTBYTE,5107,2,SUBBYTE,2553.5,PRNTBYTE,2553.5
4. ABCPrint,20060916,20060916,16:15:03,16:15:03,,3,User,MikeL,ProratePct,25,Orig_PRNTBYTE,1913,1,PRNTBYTE,478.25
5. ABCPrint,20060916,20060916,16:15:03,16:15:03,,3,User,Bill,ProratePct,25,Orig_PRNTBYTE,1913,1,PRNTBYTE,478.25
6. ABCPrint,20060916,20060916,16:15:03,16:15:03,,3,User,Mark,ProratePct,25,Orig_PRNTBYTE,1913,1,PRNTBYTE,478.25
7. ABCPrint,20060916,20060916,16:15:03,16:15:03,,3,User,Tom,ProratePct,25,Orig_PRNTBYTE,1913,1,PRNTBYTE,478.25
8. ABCPrint,20060916,20060916,16:15:03,16:15:03,,1,PrinterName,OptraL,PrinterName,OptraL,PrinterName,OptraL,PrinterName,OptraL,1,SUBBYTE,1913
10. ABCPrint,20060916,20060916,17:13:33,17:13:33,,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.
Creating Job Files

**Note** • This section assumes that you have reviewed the ITUAM Data Collectors architecture described in *ITUAM Data Collectors Architecture* beginning on page 2-4.

A job file is an XML file that specifies which ITUAM Data Collectors are run and the data collection process. ITUAM includes a sample job file that you can modify for your organization, `SampleNightly_xSeries.xml` or `SampleNightly_TDS.xml`. The sample job file that you should use depends on the ITUAM package that you are using (ITUAM [...xSeries.xml] or ITUAM for z/OS [...TDS.xml]).

**Note** • If you modify a sample job file, you need to rename the file so that it is not overwritten when you upgrade to a new version of ITUAM Data Collectors.

If you installed ITUAM Data Collectors in the default location, the sample job files are in `C:\Program Files\IBM\ITUAM\JobFiles`.

Whether you are modifying a sample job file or creating a new job file, you need to follow the structure provided `Job File Structure` on page 2-44.

**Using ITUAM 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 ITUAM 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 2-44.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNDATE</td>
<td>The current day.</td>
</tr>
<tr>
<td>CURDAY</td>
<td>The current day and the previous day.</td>
</tr>
<tr>
<td>CURWEK</td>
<td>The current week (Sun–Sat).</td>
</tr>
<tr>
<td>CURMON</td>
<td>The current month.</td>
</tr>
<tr>
<td>PREDAY</td>
<td>The previous day.</td>
</tr>
<tr>
<td>PREWEK</td>
<td>The previous week (Sun–Sat).</td>
</tr>
<tr>
<td>PREMON</td>
<td>The previous month.</td>
</tr>
</tbody>
</table>
ITUAM 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 2-4.) Depending on the collector, the LogDate parameter can be passed as a default; at the command line when running ITUAM Job Runner; or through the job file.

The LogDate parameter should be included in the job file only if you are running a snapshot collector (for example, the Windows Disk collector). Snapshot collectors collect data that is current as of the date and time that the collectors are run by ITUAM 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 can enter the LogDate parameter at the job, process, or step level of the job file 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 2-89. To provide the LogDate parameter at the step level, you need to use the Parameter element.
Using the SampleNightly_xSeries.xml Job File

If you are using the ITUAM package (not ITUAM for z/OS), you can modify the SampleNightly_xSeries.xml job file for use by your organization. The file contents are shown here. Make sure that you rename the modified file.

```xml
<?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="true"
       smtpSendJobLog="true"
       smtpServer="mail.ITUAMCustomerCompany.com"
       smtpFrom="ITUAM@ITUAMCustomerCompany.com"
       smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
       stopOnProcessFailure="false">
    <Process id="CIMSWinProcess"
             description="Process for Windows Process Collection"
             active="true">
      <Defaults>
        <Default programName="CIMSACCT"
                 accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTab1-Win.txt"/>
      </Parameters>
    </Step>
    <Step id="Server1 Collection"
          description="Server1 CIMSWinProcess"
          type="ConvertToCSR"
          programName="CIMSWinProcess\CIMSWinProcess.wsf"
          programType="wsf"
          active="true">
      <Parameters>
        <Parameter Feed="Server1"/>
      </Parameters>
    </Step>
    <Step id="Scan"
          description="Scan CIMSWinProcess"
          type="Process"
          programName="Scan"
          programType="net"
          active="true">
      <Parameters>
        <Parameter retainFileDate="false"/>
        <Parameter queueMissingFiles="false"/>
        <Parameter allowEmptyFiles="false"/>
        <Parameter useStepFiles="false"/>
      </Parameters>
    </Step>
  </Process>
</Jobs>
```

For a description of the Jobs and Job elements and attributes, see page 2-48.

For a description of the Process element and attributes, see page 2-53.

For a description of the Defaults and Default elements and attributes, see page 2-89.

For a description of the Steps and Step elements and attributes, see page 2-57.

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

For descriptions of the parameters for the Scan program, see page 2-62.
The Process step uses default parameters. To specify parameters for CIMSAcct, CIMSSort, and CIMSBill, provide a separate step for each.

For descriptions of the parameters for the DBLoad program, see page 2-76.

For descriptions of the parameters for the Cleanup program, see page 2-84.

For a descriptions of the Collectors and Collector elements, attributes, and parameters for CIMSWinDisk, see page 4-6.

```
<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>

<Process id="CIMSWinDisk"
  description="Multistep Processing for Windows Disk Collector"
  active="true">
  <Defaults>
    <Default programName="CIMSACCT"
      accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTab1-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">
      <CIMSWinDisk filename="%ProcessFolder%\CIMSWinDisk.xml"
        overwrite="true">
        <CIMSCollectors version = "1.0">
          <Collectors>
            <Collector name="CIMSWinDisk"
              instanceName="Server1-C"
              instanceDescription="Scan of Server1 C"
              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="C:\"/>
                <Parameter name="Units" value="GB"/>
                <Parameter name="NumberOfLevels" value="1"/>
              </Parameters>
            </Collector>
          </Collectors>
        </CIMSCollectors>
      </CIMSWinDisk>
    </Step>
  </Steps>
</Process>
```
Creating Job Files

```xml
<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>
```

<!-IMPORTANT NOTE: If Smart Scan is enabled in the Scan step (useStepFiles="true"), you must include a Feed parameter that exactly matches the preceding Feed parameter 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 UseCommandProcessor="false"/>
<Parameter XMLFileName="%ProcessFolder%CIMSWinDisk.xml"/>
<Parameter CollectorName="CIMSWinDisk"/>
</Parameters>
</Step>
</Step>
```
Creating Job Files

```xml
<Process id="VMware"
    description="Process for VMware Collection"
    active="true">
  <Defaults>
    <Default programName="CIMSACCT"
        acctCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTabl-Win.txt"/>
  </Defaults>
  <Steps stopOnStepFailure="true">
    <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"
        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 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>
```
Creating Job Files

```xml
<Process id="UnixOS"
    description="Process for Unix Operating System Collection"
    active="true">
    <Defaults>
        <Default programName="CIMSACCT"
            accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTable-UNIX.txt"/>
    </Defaults>
    <Steps stopOnError="true">
        <Step id="Scan"
            description="Scan Unix OS"
            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 UnixOS"
            type="Process"
            programName="SingleProcessStep"
            programType="com"
            active="true">
        </Step>
        <Step id="DatabaseLoad"
            description="Database Load for UnixOS"
            type="Process"
            programName="DBLoad"
            programType="com"
            active="true">
        </Step>
        <Step id="Cleanup"
            description="Cleanup UnixOS"
            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:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTable-UNIX.txt"/>
    </Defaults>
</Process>
```
Creating Job Files

<Steps stopOnStepFailure="true">
  <Step id="Scan"
    description="Scan Unix FS"
    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 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>
</Job>
</Jobs>
Using the SampleNightly_TDS.xml Job File

If you are using the ITUAM for z/OS package, you can modify the SampleNightly_TDS.xml job file for use by your organization. The file contents are shown here. Make sure that you rename the modified file.

For more information about z/OS data collection, see Chapter 5, Tivoli Decision Support for z/OS (TDSz) Collector.

```xml
<?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="true"
      smtpSendJobLog="true"
      smtpServer="mail.ITUAMCustomerCompany.com"
      smtpFrom="ITUAM@ITUAMCustomerCompany.com"
      smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
      stopOnProcessFailure="false">
    <Process id="IBMTDS" description="Process for Windows Process Collection"
             active="true">
      <Defaults>
        <Default programName="CIMSACCT"
                 accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\AcctTabl-zOS.txt"/>
      </Defaults>
      <Steps stopOnStepFailure="true">
        <Step id="RAFADDRLOG Collection" description="RAFADDRLOG Collection"
              type="ConvertToCSR"
              programName="IBMTDS\TDSz.wsf"
              programType="wsf"
              active="true">
          <Parameters>
            <Parameter Feed="ADDRLOG"/>
            <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafAddrDefLogDef.txt"/>
          </Parameters>
        </Step>
        <Step id="RAFBATCH Collection" description="RAFBATCH Collection"
              type="ConvertToCSR"
              programName="IBMTDS\TDSz.wsf"
              programType="wsf"
              active="true">
          <Parameters>
            <Parameter Feed="BATCH"/>
            <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafBatchDef.txt"/>
          </Parameters>
        </Step>
      </Steps>
  </Process>
</Job>
</Jobs>
```

For a description of the Jobs and Job elements and attributes, see page 2-48.

For a description of the Process element and attributes, see page 2-53.

For a description of the Defaults and Default elements and attributes, see page 2-89.

For a description of the Steps and Step elements and attributes, see page 2-57.

For a description of the valid parameters for the collection steps (type= "ConvertToCSR") in the job file, see page 5-17.
Creating Job Files

<Step id="RAFCICS Collection"
    description="RAFCICS Collection"
    type="ConvertToCSR"
    programName="IBMTDS\TDSz.wsf"
    programType="wsf"
    active="true">
  <Parameters>
    <Parameter Feed="CICS"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafCICSDef.txt"/>
  </Parameters>
</Step>

<Step id="RAFDASD Collection"
    description="RAFDASD Collection"
    type="ConvertToCSR"
    programName="IBMTDS\TDSz.wsf"
    programType="wsf"
    active="true">
  <Parameters>
    <Parameter Feed="DASD"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafDASDDef.txt"/>
  </Parameters>
</Step>

<Step id="RAFDB2 Collection"
    description="RAFDB2 Collection"
    type="ConvertToCSR"
    programName="IBMTDS\TDSz.wsf"
    programType="wsf"
    active="true">
  <Parameters>
    <Parameter Feed="DB2"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafDB2Def.txt"/>
  </Parameters>
</Step>

<Step id="RAFHSMBKUP Collection"
    description="RAFHSMBKUP Collection"
    type="ConvertToCSR"
    programName="IBMTDS\TDSz.wsf"
    programType="wsf"
    active="true">
  <Parameters>
    <Parameter Feed="HSMBKUP"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafHSMBkupDef.txt"/>
  </Parameters>
</Step>

<Step id="RAFHSMMIGR Collection"
    description="RAFHSMMIGR Collection"
    type="ConvertToCSR"
    programName="IBMTDS\TDSz.wsf"
    programType="wsf"
    active="true">
  <Parameters>
    <Parameter Feed="HSMMIGR"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafHSMMigrDef.txt"/>
  </Parameters>
</Step>
Installing ITUAM Data Collectors and Setting Up the System

Creating Job Files

<Step id="RAFIMS Collection" description="RAFIMS Collection" type="ConvertToCSR" programName="IBMTDS\TDSz.wsf" programType="wsf" active="true">
<Parameters>
<Parameter Feed="IMS"/>
<Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafIMSDef.txt"/>
</Parameters>
</Step>

<Step id="RAFJOBLOG Collection" description="RAFJOBLOG Collection" type="ConvertToCSR" programName="IBMTDS\TDSz.wsf" programType="wsf" active="true">
<Parameters>
<Parameter Feed="JOBLOG"/>
<Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafJobLogDef.txt"/>
</Parameters>
</Step>

<Step id="RAFNETSES20 Collection" description="RAFNETSES20 Collection" type="ConvertToCSR" programName="IBMTDS\TDSz.wsf" programType="wsf" active="true">
<Parameters>
<Parameter Feed="NETSES20"/>
<Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafNetSes20Def.txt"/>
</Parameters>
</Step>

<Step id="RAFNETSES4X Collection" description="RAFNETSES4X Collection" type="ConvertToCSR" programName="IBMTDS\TDSz.wsf" programType="wsf" active="true">
<Parameters>
<Parameter Feed="NETSES4X"/>
<Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafNetSes4xDef.txt"/>
</Parameters>
</Step>
Creating Job Files

For descriptions of the parameters for the Scan program, see page 2-62.
The Process step uses default parameters. To specify parameters for CIMSAcct, CIMSSort, and CIMSBill, provide a separate step for each.

For descriptions of the parameters for the DBLoad program, see page 2-76.

For descriptions of the parameters for the Cleanup program, see page 2-84.
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_xSeries.xml job file as an example, assume that the ConvertToCSR steps for CIMSWinDisk (see page 2-30) and VMware (see page 2-32) failed. All other ConvertToCSR steps in the job file ran successfully.

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

```xml
<?xml version="1.0" encoding="utf-8"?>
<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="true"
       smtpSendJobLog="true"
       smtpServer="mail.ITUAMCustomerCompany.com"
       smtpFrom="ITUAM@ITUAMCustomerCompany.com"
       smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
       stopOnProcessFailure="false">
    <Process id="CIMSWinDisk"
             description="Multistep Processing for CIMS Windows Disk Collector"
             active="true">
      <Defaults>
        <Default programName="CIMSACCT"
                 accCodeConvTable="C:\CIMS\AccountCodeTable\AccountCodeTable\AcctTabl-Win.txt"/>
        <Default logDate="RNDATE"/>
      </Defaults>
      <Steps stopOnStepFailure="true">
        <Step id="Server1 Collection"
              description="Server1 CIMSWinDisk"
              type="ConvertToCSR"
              programName="CIMSWinDisk\CIMSWinDisk.exe"
              active="true">
          <CIMSWinDisk filename="%ProcessFolder%\CIMSWinDisk.xml"
                        overwrite="true">
            <CIMSWinDisk version = "1.0"
            <Collectors>
              <Collector name="CIMSWinDisk"
                         instanceName="Server1-C"
                         instanceDescription="Scan of Server1 C"
                         active="True">
            <Parameters>
              <Parameter name="LogDate" value="%RNDATE%" />
              <Parameter name="Retention" value="KEEP" />
          </Step>
        </Process>
      </Steps>
    </Job>
  </Jobs>
```

Here you reprocess the CIMSWinDisk ConvertToCSR step that failed.
<Parameter name="Feed" value="Server1-C" />
<Parameter name="OutputFolder" value="%ProcessFolder%" />
<Parameter name="PathToScan" value="C:" />
<Parameter name="Units" value="GB" />
<Parameter name="NumberOfLevels" value="1" />
</Parameters>
</Collector>
</Collectors>
</CIMSCollectors>
</CIMSWinDisk>
</Parameters>

<!--IMPORTANT NOTE: If Smart Scan is enabled in the Scan step (useStepFiles="true"), you must include a Feed parameter that exactly matches the preceding Feed parameter or include the scanFile="<file name>" parameter that specifies the path and file name of the file to be scanned.-->
<Parameter Feed="Server1-C" />
<!--Parameter scanFile=""/-->
<Parameter UseStandardParameters="false" />
<Parameter UseCommandProcessor="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="true" />
</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>

The Feed or scanFile attribute is required when processing CIMWinDisk. See page 2-43.

The useStepFiles="true" attribute enables Smart Scan. See page 2-43.
Creating Job Files

Here you reprocess the VMware ConvertToCSR step that failed.

<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="VMware"
description="Process for VMware Collection"
joblogShowStepOutput="true"
joblogShowStepParameters="true"
active="true">
  <Defaults>
    <Default programName="CIMSACCT"
      accCodeConvTable="C:\CIMS\AccountCodeTable\AccTabl-Win.txt"/>
  </Defaults>
  <Steps stopOnStepFailure="true">
    <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">
      <Parameters>
        <Parameter retainFileDate="false"/>
        <Parameter allowMissingFiles="false"/>
        <Parameter allowEmptyFiles="false"/>
        <Parameter useStepFiles="true"/>
      </Parameters>
    </Step>
    <Step id="Process"
description="Standard Processing for VMware"
type="Process"
programName="SingleProcessStep"
programType="com"
active="true">
    </Step>
  </Steps>
</Process>
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, `VMware\Server1\20060916.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 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 of the CSR or CSR+ file to be scanned as shown in the preceding job file example. For a description of the `scanFile` parameter, see page 2-88.
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 ITUAM 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 2-3 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processFolder</td>
<td>Optional</td>
<td>In most cases, you will not need to use this attribute. By default, the path to the Processes folder set in the CIMSCfgOptions table is used. This attribute is required only if you are collecting data on a computer other than the central ITUAM 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 &quot;pull&quot; CSR files from the other computer to the central server for processing.</td>
</tr>
<tr>
<td>smtpSendJobLog</td>
<td>Optional</td>
<td>Specifies whether the job log should be sent via e-mail. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;true&quot; (send via e-mail)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;false&quot; (do not send)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

Table 2-3 • Jobs Element Attributes
Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtpServer</td>
<td>Optional</td>
<td>The name of the SMTP mail server that will be used to send the job log. The default is &quot;mail.ITUAMCustomerCompany.com&quot;.</td>
</tr>
<tr>
<td>smtpFrom</td>
<td>Optional</td>
<td>The fully qualified e-mail address of the e-mail sender. The default is &quot;<a href="mailto:ITUAM@ITUAMCustomerCompany.com">ITUAM@ITUAMCustomerCompany.com</a>&quot;.</td>
</tr>
</tbody>
</table>

Table 2-3 • Jobs Element Attributes (Continued)
Creating Job Files

smtpTo

Optional

The fully qualified e-mail address of the e-mail receiver. The syntax for an address defined by this attribute can be any of the following.

- user@domain

  Example: jsmith@xyzco.com
  When this syntax is used, the default mail server is the server defined by the smtpServer attribute.

- servername:user@domain

  Example:
  mail.xyzco.com:jsmith@xyzco.com
  When the servername: syntax is used, the mail server specified for the attribute overrides the server defined by the smtpServer attribute.

- servername:userID:password:user@domain

  Example:
  mail.xyzco.com:janes:global:jsmith@xyzco.com
  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".

  The default is "John.ITUAMUser@ITUAMCustomerCompany.com".

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtpTo</td>
<td>Optional</td>
<td>The fully qualified e-mail address of the e-mail receiver. The syntax for an address defined by this attribute can be any of the following.</td>
</tr>
</tbody>
</table>

Table 2-3 • Jobs Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtpSubject</td>
<td>Optional</td>
<td>The text that you want to appear in the e-mail subject. If you do not provide a subject, the following appears: ITUAM job &lt;job name&gt; running on &lt;server name&gt; completed &lt;successfully or with x warning(s)/with x error(s)&gt;</td>
</tr>
<tr>
<td>smtpBody</td>
<td>Optional</td>
<td>The text that you want to appear in the e-mail body. If you do not provide body text, the following appears: Attached are results from a JobRunner execution.</td>
</tr>
</tbody>
</table>

Table 2-3  • Jobs Element Attributes (Continued)
Creating Job Files

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 2-4 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Required</td>
<td>A text string name for the job. This value must be unique from other job ID values in the file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>id=&quot;Nightly&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In this example, the subfolder that contains log files for this job will also be named <strong>Nightly</strong>. See <strong>Job Log Files (LogFiles Folder)</strong> on page 2-9.</td>
</tr>
<tr>
<td>description</td>
<td>Optional</td>
<td>A text string description of the job (maximum of 255 characters).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>description=&quot;Nightly collection and processing&quot;</td>
</tr>
<tr>
<td>active</td>
<td>Optional</td>
<td>Specifies whether the job should be run. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;true&quot; (run the job)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;false&quot; (do not run the job)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
</tbody>
</table>

Table 2-4  Job Element Attributes
### Creating Job Files

#### dataSourceId
- **Required or Optional:** Optional
- **Description:** The ITUAM Data Source for the ITUAM database.
  - **Example:**
    ```
    dataSourceId=CSDev
    ```
  - If this parameter is not provided, the ITUAM Data Source that is set as the Web/collector default in the ITUAM Data Source Maintenance dialog box in ITUAM Administrator is used.

  To use a ITUAM Data Source other than the default, set this parameter to the appropriate ITUAM Data Source ID.

  For more information about creating and using ITUAM Data Sources, refer to the *IBM Tivoli Usage and Accounting Manager Administrator’s Guide*.

#### joblogShowStepParameters
- **Required or Optional:** Optional
- **Description:** Specifies whether parameters for the steps in a job are written to the job log file.
  - **Valid values:**
    - "true" (parameters are written to the job log)
    - "false" (parameters are not written)

  **Example:**
  ```
  joblogShowStepParameters=true
  ```

  The default is "true".

#### joblogShowStepOutput
- **Required or Optional:** Optional
- **Description:** Specifies whether output generated by the steps in a job is written to the job log file.
  - **Valid values:**
    - "true" (step output is written to the job log)
    - "false" (step output is not written)

  **Example:**
  ```
  joblogShowStepOutput=true
  ```

  The default is "true".

---

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSourceId</td>
<td>Optional</td>
<td>The ITUAM Data Source for the ITUAM database.</td>
</tr>
</tbody>
</table>
|                           |                      | **Example:**
|                           |                      | `dataSourceId=CSDev`                                                       |
|                           |                      | If this parameter is not provided, the ITUAM Data Source that is set as     |
|                           |                      | the Web/collector default in the ITUAM Data Source Maintenance dialog box   |
|                           |                      | in ITUAM Administrator is used.                                             |
|                           |                      | To use a ITUAM Data Source other than the default, set this parameter to    |
|                           |                      | the appropriate ITUAM Data Source ID.                                      |
|                           |                      | For more information about creating and using ITUAM Data Sources, refer to  |
|                           |                      | the *IBM Tivoli Usage and Accounting Manager Administrator’s Guide*.       |
| joblogShowStepParameters   | Optional             | Specifies whether parameters for the steps in a job are written to the job |
|                           |                      | log file.                                                                   |
|                           |                      | **Valid values:**                                                           |
|                           |                      | - "true" (parameters are written to the job log)                            |
|                           |                      | - "false" (parameters are not written)                                      |
|                           |                      | The default is "true".                                                     |
| joblogShowStepOutput      | Optional             | Specifies whether output generated by the steps in a job is written to     |
|                           |                      | the job log file.                                                           |
|                           |                      | **Valid values:**                                                           |
|                           |                      | - "true" (step output is written to the job log)                            |
|                           |                      | - "false" (step output is not written)                                     |
|                           |                      | The default is "true".                                                     |
Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processFolder</td>
<td>Optional</td>
<td>In most cases, you will not need to use this attribute. By default, the path to the Processes folder set in the CIMSCfgOptions table is used. This attribute is required only if you are collecting data on a computer other than the central ITUAM 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 &quot;push&quot; CSR files from the other computer to the central server for processing.</td>
</tr>
<tr>
<td>processPriorityClass</td>
<td>Optional</td>
<td>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, the use of the Low or BelowNormal value is recommended. These values allow other processes (for example, IIS and SQL Server tasks) to take precedence. Consult IBM before using a value other than Low or BelowNormal. Note: 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.</td>
</tr>
<tr>
<td>joblogWriteToTextFile</td>
<td>Optional</td>
<td>Specifies whether the job log should be written to a text file. Valid values are: &quot;true&quot; (writes to a text file) and &quot;false&quot; (does not write to a text file). The default is &quot;true&quot;.</td>
</tr>
<tr>
<td>joblogWriteToXMLFile</td>
<td>Optional</td>
<td>Specifies whether the job log should be written to an XML file. Valid values are: &quot;true&quot; (writes to an XML file) and &quot;false&quot; (does not write to an XML file). The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>joblogWriteToDB</td>
<td>Optional</td>
<td>Specifies whether the job log should be written to the ITUAM database. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (writes to the database)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data is written to the CIMSJobLog tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (does not write to the database)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td>smtpSendJobLog</td>
<td>Optional</td>
<td>Specifies whether the job log should be sent via e-mail. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (send via e-mail)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (do not send)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td>smtpServer</td>
<td>Optional</td>
<td>The name of the SMTP mail server that will be used to send the job log.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;mail.ITUAMCustomerCompany.com&quot;.</td>
</tr>
<tr>
<td>smtpFrom</td>
<td>Optional</td>
<td>The fully qualified e-mail address of the e-mail sender.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;<a href="mailto:ITUAM@ITUAMCustomerCompany.com">ITUAM@ITUAMCustomerCompany.com</a>&quot;.</td>
</tr>
<tr>
<td>smtpTo</td>
<td>Optional</td>
<td>The fully qualified e-mail address of the e-mail receiver. See the description on page 2-46.</td>
</tr>
<tr>
<td>smtpSubject</td>
<td>Optional</td>
<td>The text that you want to appear in the e-mail subject.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you do not provide a subject, the following appears:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITUAM job &lt;job name&gt; running on &lt;server name&gt; completed &lt;successfully or with x warning(s)/with x error(s)&gt;</td>
</tr>
</tbody>
</table>

Table 2-4 • Job Element Attributes (Continued)
## Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtpBody</td>
<td>Optional</td>
<td>The text that you want to appear in the e-mail body. If you do not provide body text, the following appears: Attached are results from a JobRunner execution.</td>
</tr>
<tr>
<td>stopOnProcessFailure</td>
<td>Optional</td>
<td>Specifies whether a job with multiple processes should stop if any of the processes fail. Valid values are: &quot;true&quot; (stop processing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;false&quot; (continue processing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If stopOnStepFailure is set to &quot;false&quot; at the Steps element level in a process, processing continues regardless of the value set for stopOnProcessFailure.</td>
</tr>
</tbody>
</table>

Table 2-4 • 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 (VMware, Windows process, UNIX/Linux filesystem, etc.).

Table 2-5 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Required</td>
<td>A text string name for the process. This value must be unique from the other process ID values in the job. This value must match the name of a process definition folder for a collector in the Processes folder (see Process Definitions (Processes Folder) on page 2-13). If the buildProcessFolder attribute is not included or is set to &quot;true&quot; (the default), ITUAM Job Runner will create a process definition folder of the same name in the Processes folder if the process definition folder does not exist. Example: id=&quot;ABCSoftware&quot;</td>
</tr>
<tr>
<td>description</td>
<td>Optional</td>
<td>A text string description of the process (maximum of 255 characters). Example: description=&quot;Process for ABCSoftware&quot;</td>
</tr>
</tbody>
</table>

Table 2-5  •  Process Element Attributes
### Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| buildProcessFolder         | Optional             | Specifies whether ITUAM Job Runner will create a process definition folder with the same name as the id attribute value in the Processes folder. If you are using ITUAM Job Runner to perform data collection, a process folder is always required. If you do not include this attribute or set it to "true", a process definition folder is created automatically if it does not already exist. This attribute is only applicable if you are using ITUAM Job Runner to run a script or program that does not require a process definition folder. For example, you can use ITUAM Job Runner to run the ReportDistribution.wsf script used for batch reporting as described in the *IBM Tivoli Usage and Accounting Manager Administrator’s Guide*. Valid values are:  
  - "true" (the process definition folder is created)  
  - "false" (the process definition folder is not created)  
  The default is "true". |
| joblogShowStepParameters   | Optional             | Specifies whether parameters for the steps in a process are written to the job log file. Valid values are:  
  - "true" (parameters are written to the job log)  
  - "false" (parameters are not written)  
  The default is "true". |

Table 2-5 • Process Element Attributes (Continued)
### Attribute | Required or Optional | Description
--- | --- | ---
joblogShowStepOutput | Optional | Specifies whether output generated by the steps in a process is written to the job log file. Valid values are:
- "true" (step output is written to the job log)
- "false" (step output is not written)
  
  The default is "true".
  
processPriorityClass | Optional | This attribute determines the priority in which the process is run. Valid values are: Low, BelowNormal (the default), Normal, AboveNormal, and High. Because a job can use a large amount of CPU time, the use of the Low or BelowNormal value is recommended. These values allow other processes (for example, IIS and SQL Server tasks) to take precedence. Consult IBM before using a value other than Low or BelowNormal.

  **Note:** 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.

active | Optional | Specifies whether the process should be run. Valid values are:
- "true" (run the process)
- "false" (do not run the process)

  The default is "true".

*Table 2-5  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 2-6.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stopOnStepFailure</td>
<td>Optional</td>
<td>Specifies whether processing should continue if any of the active steps in the process fail. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;true&quot; (processing fails)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the stopOnProcessFailure attribute is also set to &quot;true&quot;, the remaining processes in the job are not executed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If stopOnProcessFailure is set to &quot;false&quot;, the remaining processes in the job are executed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;false&quot; (processing continues)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In this situation, all remaining processes in the job are also executed regardless of the value set for stopOnProcessFailure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
</tbody>
</table>

Table 2-6 • 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 2-7 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Required</td>
<td>A text string name for the step. This value must be unique from other step ID values in the process.</td>
</tr>
<tr>
<td>description</td>
<td>Optional</td>
<td>A text string description of the step (maximum of 255 characters).</td>
</tr>
<tr>
<td>active</td>
<td>Optional</td>
<td>Specifies whether the step should be run. Valid values are:</td>
</tr>
<tr>
<td>type</td>
<td>Required</td>
<td>The type of step that is being implemented: &quot;ConvertToCSR&quot; or &quot;Process&quot;.</td>
</tr>
</tbody>
</table>

- "true" (run the step)
- "false" (do not run the step)

The default is "true".

"ConvertToCSR" specifies that the step performs data collection and conversion and creates a CSR file.

Process specifies that the step executes one of the programs described in ITUAM Processing Programs (Process Engine\JobLibrary Folder) on page 2-10.

Table 2-7 • Step Element Attributes
### Creating Job Files

**Attribute**

<table>
<thead>
<tr>
<th>programName</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>programName</td>
<td>Required</td>
<td>The name of the program that will be run by the step.</td>
</tr>
</tbody>
</table>

If the `type` attribute is `ConvertToCSR` and the `programType` attribute is `wsf`, this value can be either of the following:

- The path and name of a conversion script in the Collectors folder. For example, "VMware\VMware.wsf" specifies the conversion script for the VMware collector.
- 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`.

If the `type` attribute is `ConvertToCSR` and the `programType` attribute is `console`, this value can be the full path or just the name of console application (make sure that you include the file extension, e.g., `CIMSPRAT.exe`).

If you do not include the path, ITUAM Job Runner searches the Collectors, Process Engine, and Scripts folders for the program.

If the `type` attribute is `Process`, this value is the name of an ITUAM program (e.g., "Scan", "CIMSACCT", "CIMSBILL", "DBLoad", etc).

**Examples:**

- `programName="VMware\Vmware.wsf"`
- `programName="CIMSWinDisk.exe"
- `programName="Cleanup"`

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

Table 2-7 • Step Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processPriorityClass</td>
<td>Optional</td>
<td>This attribute determines the priority in which the step is run. Valid values are: Low, BelowNormal (the default), Normal, AboveNormal, and High. Because a job can use a large amount of CPU time, the use of the Low or BelowNormal value is recommended. These values allow other processes (for example, IIS and SQL Server tasks) to take precedence. Consult IBM before using a value other than Low or BelowNormal. Note: A priority of Low or BelowNormal will not cause the step to run longer if the system is idle. However, if other tasks are running, the step will take longer.</td>
</tr>
</tbody>
</table>
| programType                      | Optional             | The type of program specified by the programName attribute:  
  ■ "wsf"—Windows Scripting File  
  ■ "ce"—ITUAM Conversion Engine  
  ■ "console"—Console Application  
  ■ "com"—COM Component  
  ■ "net"—.Net Component  
  ■ "java"—Java application  
  The default is "net". |
| joblogShowStepParameters         | Optional             | Specifies whether parameters for the step are written to the job log file. Valid values are:  
  ■ "true" (parameters are written to the job log)  
  ■ "false" (parameters are not written)  
  The default is "true". |

Table 2-7 • Step Element Attributes (Continued)
Creating Job Files

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 6.

The valid attributes for process step parameters (type=Process) are listed in Table 2-8 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 2-86.

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 CIMSAcct and CIMSBill, do not include the path with the file name. These files should reside in the collector’s process definition folder.

The exceptions are the account code conversion table used by CIMSAcct (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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>joblogShowStepOutput</td>
<td>Optional</td>
<td>Specifies whether output generated by the step is written to the job log file. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (step output is written to the job log)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (step output is not written)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
</tbody>
</table>

Table 2-7  Step Element Attributes (Continued)
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 CIMSConfigOptions 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.
## Valid Parameters by Program Name

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan</td>
<td>retainFileDate</td>
<td>Optional</td>
<td>Specifies whether the date is retained in the final CSR file (i.e., yyyymmdd.txt rather than CurrentCSR.txt). Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;true&quot; (the file name is yyyymmdd.txt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;false&quot; (the file name is CurrentCSR.txt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td>allowMissingFiles</td>
<td>Optional</td>
<td>Specifies whether a warning or error occurs when feed subfolders do not contain a file that matches the log date value. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;true&quot; (a warning occurs, processing continues)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;false&quot; (an error occurs, processing fails)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td>allowEmptyFiles</td>
<td>Optional</td>
<td>Specifies whether a warning or error occurs when feed subfolders contain a zero-length file that matches the log date value. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;true&quot; (a warning occurs, processing continues)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;false&quot; (an error occurs, processing fails)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

### Table 2-8 • Parameter Element Attributes
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan (continued)</td>
<td>excludeFile</td>
<td>Optional</td>
<td>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. <strong>Example:</strong> <code>excludeFile=&quot;MyCSR*&quot;</code> In this example, all files that begin with MyCSR are not scanned.</td>
</tr>
<tr>
<td></td>
<td>excludeFolder</td>
<td>Optional</td>
<td>The name of a feed subfolder 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 page 2-14). <strong>Example:</strong> <code>excludeFolder=&quot;Server1&quot;</code> In this example, the feed subfolder Server1 is not scanned.</td>
</tr>
<tr>
<td></td>
<td>includeFile</td>
<td>Optional</td>
<td>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. <strong>Example:</strong> <code>includeFile=&quot;MyCSR.txt&quot;</code> In this example, files in the feed subfolders that are named MyCSR are scanned.</td>
</tr>
</tbody>
</table>

**Table 2-8 • Parameter Element Attributes (Continued)**
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

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

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| Scan (continued)     | useStepFiles           | Optional             | Specifies whether the Smart Scan feature is enabled (see page 2-40 for a description of this feature). Valid values are:  
  - "true" (Smart Scan is enabled)  
  - "false" (Smart Scan is not enabled)  
  The default is "false".  
  By default, Smart Scan looks for a file named LogDate.txt in the process definition feed subfolders (e.g., CIMSWinProcess/Server1/20060624.txt). If you want to override the default name, use the parameter attribute scanFile in the conversion step (see page 2-88). |

| CIMSPRAT             | XMLFileName            | Required             | The path and the name of the XML file containing the proration parameters used by CIMSPrat.  
  For a description of this program, see page 2-11.  
  For an example of the parameters for this program in a job file, see the Sample Nightly.xml file.  
  useStandardParameters and useCommandProcessor | Optional | For descriptions of these attributes, see page 2-86. |
Creating Job Files

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

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSACCT</td>
<td>inputFile</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>inputFile</td>
<td></td>
<td>inputFile=&quot;MyCSR.txt&quot;</td>
</tr>
</tbody>
</table>
|                      |                |                      | The default is "CurrentCSR.txt".

**Note:** 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 *IBM Tivoli Usage and Accounting Manager Administrator’s Guide*.

|                      | detailFile     | Optional             | The name of the CIMSAcct Detail file. This file must be in the collector’s process definition folder—do not include a path.             |
|                      |                |                      | **Example:**                                                                                                                                |
|                      | detailFile     |                      | detailFile="MyDetail.txt"                                                                                                                                 |
|                      |                |                      | The default is "Detail.txt".

|                      | accCodeConvTable | Optional | The name of account code conversion table used by CIMSAcct. Include a path if the table is in a location other than the collector’s process definition folder. |
|                      |                |          | **Examples:**                                                                                                                              |
|                      | accCodeConvTable |          | accCodeConvTable= "MyAcctTbl.txt"                                                                                                          |
|                      |                |          | accCodeConvTable= "E:\Processes\Account\"                                             |
|                      |                |          | "MyAcctTbl.txt"                                                                       |
|                      |                |          | The default is "AcctTbl.txt".                                                           |
## Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSACCT (continued)</td>
<td>resultsFile</td>
<td>Optional</td>
<td>The name of the CIMSAcct Results file. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>resultsFile= &quot;MyAcctResults.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The <strong>default</strong> is &quot;AcctResults.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>controlFile</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>controlFile= &quot;MyAcctCntl.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The <strong>default</strong> is &quot;AcctCntl.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>messageFile</td>
<td>Optional</td>
<td>The name of the CIMSAcct Message file. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>messageFile= &quot;MyAcctMsg.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The <strong>default</strong> is &quot;AcctMsg.txt&quot;</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Creating Job Files

#### Table 2-8  Parameter Element Attributes (Continued)

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSACCT (continued)</td>
<td>exceptionFile</td>
<td>Optional</td>
<td>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. The file name should contain the log date so that it is not overwritten when CIMSACct is run again. Example: exceptionFile=&quot;Exception_%LogDate_End%.txt&quot; The default is &quot;Exception.txt&quot;.</td>
</tr>
<tr>
<td></td>
<td>identFile</td>
<td>Optional</td>
<td>The name of the Ident file. This file must be in the collector’s process definition folder—do not include a path. Example: identFile=&quot;MyIdent.txt&quot; The default is &quot;Ident.txt&quot;.</td>
</tr>
</tbody>
</table>
|                      | createDBInf | Optional             | Specifies whether the ODBCINF.txt file should be generated. This file is required by the Fujitsu COBOL Workstation Run-time program required by ITUAM Processing Engine. Valid values are:  
  - "true" (the file is generated)  
  - "false" (the file is not generated)  
  The default is "true". |


Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| CIMSACCT (continued) | controlCard       | Optional             | A valid CIMSAcct control statement or statements. All CIMSAcct control statements are stored in the CIMSAcct control file (see page 2-66). **Note:** 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. To define multiple control statements, you need to use a separate parameter for each statement. **Example:**

```xml
<Parameter controlCard="TEST A" />
<Parameter controlCard="VERIFY DATA ON" />
```

|                  | logMessageFileOutput | Optional | Specifies whether the text of the CIMSAcct Message file is included in the job log file. Valid values are:
|                  |                    |          | - "true" (the text is included)
|                  |                    |          | - "false" (the text is not included)
|                  |                    |          | The default is "true". |

|                  | logResultFileOutput | Optional | Specifies whether the text of the CIMSAcct Results file is included in the job log file. Valid values are:
|                  |                    |          | - "true" (the text is included)
|                  |                    |          | - "false" (the text is not included)
|                  |                    |          | The default is "true". |

Table 2-8 • Parameter Element Attributes (Continued)
### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSACCT (continued)</td>
<td>createCSRPFi1e</td>
<td>Optional</td>
<td>Specifies whether a CSR+ file is written. CSR+ files are the same as CSR files except the file records contain an additional record header. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;true&quot; (CSR+ is written)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;false&quot; (CSR+ is not written)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td>CSRPFi1e</td>
<td>Optional</td>
<td>Specifies the name of the CSR+ file that will be written.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CSRPFi1e=&quot;MyCSRPFi1e.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;CSRPFi1e.txt&quot;.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
## Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSSORT</td>
<td>inputFilename</td>
<td>Optional</td>
<td>The file name for the CIMSAcct Detail file. The path is required only if the file is not in the process definition folder. Examples: inputFilename=&quot;Detail.txt&quot; inputFilename=&quot;E:\FolderA\Detail.txt&quot; The default is &lt;default process definition folder path&gt; +&quot;Detail.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>outputFilename</td>
<td>Optional</td>
<td>The path and file name for the sorted output. The path is required only if the file is not sent to the process definition folder. Examples: outputFilename=&quot;Detail.txt&quot; outputFilename=&quot;E:\FolderA\Detail.txt&quot; The default is &lt;default process definition folder path&gt; +&quot;Detail.txt&quot;</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Creating Job Files

#### Program Name or Type

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSBILL</td>
<td>detailFileIn</td>
<td>Optional</td>
<td>The name of the input CIMSAcct Detail, CIMSBill Detail, or Summary file to be processed. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td>detailFileOut</td>
<td>Optional</td>
<td>The name of the CIMSBill Detail file produced. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td>summaryFile</td>
<td>Optional</td>
<td>The name of the Summary file produced. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
</tbody>
</table>

**Example:**

detailFileIn= "MyDetail.txt"

The default is "Detail.txt".

detailFileOut= "MyBillDetail.txt"

The default is "BillDetail.txt".

summaryFile= "MyBillSummary.txt"

The default is "BillSummary.txt".

Table 2-8 • Parameter Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSBILL (continued)</td>
<td>resultsFile</td>
<td>Optional</td>
<td>The name of the CIMSBill Results file. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> resultsFile= &quot;MyBillResults.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;BillResults.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>controlFile</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> controlFile= &quot;MyBillCntl.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;BillCntl.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>messageFile</td>
<td></td>
<td>The name CIMSBill Message file. This file must be in the collector's process definition folder—do not include a path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> messageFile= &quot;MyBillMsg.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;BillMsg.txt&quot;</td>
</tr>
<tr>
<td></td>
<td>multTableFile</td>
<td>Optional</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Examples:</strong> multTableFile= &quot;MyMultTable.txt&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>multTableFile= &quot;E:\Processes\Prorate\MyMultTable.txt&quot;</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Developing the Job Files

CIMSBill (continued)

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| CIMSBill             | createDBInf| Optional             | Specifies whether the ODBCINF.txt file should be generated. This file is required by the Fujitsu COBOL Workstation Run-time program required by ITUAM Processing Engine. Valid values are:  
  - "true" (the file is generated)  
  - "false" (the file is not generated)  
The default is "true". |
|                      | dateSelection| Optional            | Defines a date range for records to be processed by CIMSBill. Valid values are a from and to date range in yyyymmdd format or a ITUAM date keyword.  
**Examples:**  
dateSelection="20060117 20060118"  
In this example, CIMSBill will process records with an accounting end dates of January 17 and 18, 2006.  
dateSelection="PREDAY"  
In this example, CIMSBill will process records with an accounting end date one day prior to the date ITUAM Job Runner is run. For more information about accounting dates, refer to the IBM Tivoli Usage and Accounting Manager Administrator’s Guide. |

Table 2-8  •  Parameter Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSBILL (continued)</td>
<td>reportDate</td>
<td>Optional</td>
<td>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 yyyymmdd format or a ITUAM date keyword. 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 &quot;CURMON&quot;, which sets the accounting start date to the first of the month and the end date to the last day of the month.</td>
</tr>
<tr>
<td>controlCard</td>
<td>controlCard</td>
<td>Optional</td>
<td>A valid CIMSBill control statement. All CIMSBill control statements are stored in the CIMSBill control file (see page 2-72). Note: 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. To define multiple control statements, you need to use a separate parameter for each statement. Example:</td>
</tr>
</tbody>
</table>

```xml
<Parameter controlCard= "CLIENT SEARCH ON" />  
<Parameter controlCard= "DEFINE J1 1 1" />
```

Table 2-8 • Parameter Element Attributes (Continued)
## Installing ITUAM Data Collectors and Setting Up the System

### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSBILL (continued)</td>
<td>logMessageFileOutput</td>
<td>Optional</td>
<td>Specifies whether the text of the CIMSBill Message file is included in the job log file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;true&quot; (the text is included)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;false&quot; (the text is not included)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
<tr>
<td></td>
<td>logResultFileOutput</td>
<td>Optional</td>
<td>Specifies whether the text of the CIMSBill Results file is included in the job log file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;true&quot; (the text is included)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;false&quot; (the text is not included)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| DBLoad               | loadType  | Required if you want to load a specific file rather than 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:  
  - Summary  
  - BillDetail (Detail file produced by CIMSBill)  
  - AcctDetail (Detail file produced by CIMSAcct)  
  - Ident  
  For more information about these file types, refer to the IBM Tivoli Usage and Accounting Manager Administrator’s Guide. |

For a description of this program, see page 2-11.

<table>
<thead>
<tr>
<th></th>
<th>filename</th>
<th>Required if loadType attribute is used</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Required if loadType attribute is used</td>
<td>The default file names are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required if loadType attribute is used</td>
<td>BillSummary.txt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required if loadType attribute is used</td>
<td>BillDetail.txt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required if loadType attribute is used</td>
<td>Ident.txt or Ident_yyymmdd.txt</td>
</tr>
</tbody>
</table>

Example:

file="BillSummary.txt"

Table 2-8 • Parameter Element Attributes (Continued)
Creating Job Files

DBLoad

**allowDetailDuplicates**

Optional

Specifies whether duplicate Detail files can be loaded into the database. Valid values are:

- "true" (duplicate loads can be loaded)
- "false" (duplicate loads cannot be loaded)

The default is "false".

**allowSummaryDuplicates**

Optional

Specifies whether duplicate Summary files can be loaded into the database. Valid values are:

- "true" (duplicate loads can be loaded)
- "false" (duplicate loads cannot be loaded)

The default is "false".

**useBulkLoad**

Optional

Specifies whether the SQL Server bulk load facility should be used to improve load performance. Valid values are:

- "true" (bulk load is used)
- "false" (bulk load is not used)

The default is "true".

**useDatedFiles**

Optional

If set to "true", only files that contain a date matching the LogDate parameter value are loaded into the database. The default is "false".

---

**Table 2-8 • Parameter Element Attributes (Continued)**
### WaitFile

For a description of this program, see page 2-12.

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaitFile</td>
<td>pollingInterval</td>
<td>Optional</td>
<td>The number of seconds to check for file availability (maximum of 10,080 [one week]).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> pollingInterval=&quot;60&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This example specifies a polling interval of 60 seconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is 5 seconds.</td>
</tr>
<tr>
<td></td>
<td>timeout</td>
<td>Optional</td>
<td>The number of seconds that ITUAM Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> timeout=&quot;18000&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This example specifies a timeout of 5 hours.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is to wait indefinitely.</td>
</tr>
<tr>
<td></td>
<td>timeoutDateTime</td>
<td>Optional</td>
<td>A date and time up to which ITUAM Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails. The date and time must be in the format yyyymmdd hh:mm:ss.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> timeoutDateTime=&quot;%rndate% 23:59:59&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This example specifies a timeout of 23:59:59 on the day ITUAM Job Runner is run.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is to wait indefinitely.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaitFile (continued)</td>
<td>filename</td>
<td>Required</td>
<td>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. If the file contains a date, include a variable string for the date. Example: filename=&quot;BillSummary_%LogDate_End%.txt&quot; In this example, ITUAM Job Runner will wait for Summary files that contain the same end date as the %LogDate_End% value (see page 2-61).</td>
</tr>
<tr>
<td>FileTransfer</td>
<td>continueOnError</td>
<td>Optional</td>
<td>For a multi-file transfer, specifies whether subsequent file transfers continue if a transfer fails. Valid values are: &quot;true&quot; (file transfer continues) &quot;false&quot; (file transfer does not continue) The default is &quot;false&quot;.</td>
</tr>
<tr>
<td>type</td>
<td>Required</td>
<td></td>
<td>The type of file transfer. Valid values are: &quot;ftp&quot; (File Transfer Protocol [FTP] transfer) &quot;Windows&quot; (Windows transfer)</td>
</tr>
</tbody>
</table>

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.

**Table 2-8  •  Parameter Element Attributes (Continued)**
Creating Job Files

FileTransfer (continued)

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileTransfer</td>
<td>from and to</td>
<td>Required</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- type=&quot;ftp&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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 page 2-83).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>from=&quot;ftp:///LogFiles/%LogDate_End%.log&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to=&quot;file://\Server1\LogFiles%LogDate_End%.log&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>from=&quot;file://\Server1\LogFiles%LogDate_End%.log&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to=&quot;ftp:///LogFiles/%LogDate_End%.log&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note that the use of a UNC is recommended for the file:// path as shown in these examples.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The from and to file names can be different.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a description of the %LogDate_End% variable, see page 2-61.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| FileTransfer        | from and to | Required | You can include the URL prefix file:/// before the from and to file paths or leave it off. The use of a UNC path is recommended as shown in the following example:  
from="file://\\Server1\LogFiles\%LogDate_End%.log"  
to="file://\\Server2\LogFiles\%LogDate_End%.log"  
The from and to file names can be different.  
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:  
from="\\Server1\LogFiles\%LogDate_End%*.log"  
to="\\Server2\LogFiles" |

### Table 2-8 • Parameter Element Attributes (Continued)
Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileTransfer</td>
<td>overwrite</td>
<td>Optional</td>
<td>Specifies whether the destination file is overwritten. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;true&quot; (the file is overwritten)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ &quot;false&quot; (the file is not overwritten)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

The following attributes are for FTP transfer only.

<table>
<thead>
<tr>
<th>connectionType</th>
<th>Optional</th>
<th>Describes how the connection address is resolved. This is an advanced configuration option that should be used only after consulting IBM. Valid values are:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>■ &quot;PRECONFIG&quot; (retrieves the proxy or direct configuration from the registry)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;DIRECT&quot; (resolves all host names locally)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;NOAUTOPROXY&quot; (retrieves the proxy or direct configuration from the registry and prevents the use of a startup Microsoft JScript or Internet Setup (INS) file)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;PROXY&quot; (passes requests to the proxy unless a proxy bypass list is supplied and the name to be resolved bypasses the proxy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;PRECONFIG&quot;.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Installing ITUAM Data Collectors and Setting Up the System

#### FileTransfer (continued)

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileTransfer</td>
<td>passive</td>
<td>Optional</td>
<td>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 IBM.</td>
</tr>
<tr>
<td>proxyServerBypass</td>
<td>Optional</td>
<td></td>
<td>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 options is used only when connectionType=&quot;PROXY&quot;. This is an advanced configuration option that should be used only after consulting IBM.</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td></td>
<td>If connectionType=&quot;PROXY&quot;, the name of the proxy server(s) to use. This is an advanced configuration option that should be used only after consulting IBM.</td>
</tr>
<tr>
<td>serverName</td>
<td>Required</td>
<td></td>
<td>A valid FTP IP address or server name. Example: <code>serverName=&quot;ftp.xyzco.com&quot;</code></td>
</tr>
</tbody>
</table>

---

**Table 2-8 • Parameter Element Attributes (Continued)**
### Creating Job Files

#### FileTransfer

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transferType</td>
<td>Optional</td>
<td>The type of file transfer. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;binary&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;ascii&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;binary&quot;.</td>
</tr>
<tr>
<td>userId</td>
<td>Optional</td>
<td>The user ID used to log on to the FTP server.</td>
</tr>
<tr>
<td>userPassword</td>
<td>Optional</td>
<td>The user password used to log on to the FTP server.</td>
</tr>
</tbody>
</table>

#### Cleanup

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder</td>
<td>Optional</td>
<td>By default, the Cleanup program deletes files with file names containing the date in yyyymmdd format from the collector's process definition folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If you want to delete files from another folder, use this attribute to specify the path and folder name.</td>
</tr>
<tr>
<td>daysToRetainFiles</td>
<td>Optional</td>
<td>The number of days that you want to keep the yyyymmdd files after their creation date.</td>
</tr>
</tbody>
</table>

**Example:**

```plaintext
folder="\Server1\LogFiles"
```

**Example:**

```plaintext
daysToRetainFiles="60"
```

This example specifies that all files that are older than 60 days from the current date are deleted.

The default is 45 days from the current date.

---

**Table 2-8 • Parameter Element Attributes (Continued)**
### Creating Job Files

#### Cleanup (continued)

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleanup</td>
<td>dateToRetainFiles</td>
<td>Optional</td>
<td>A date by which all yyyyymmdd files that were created prior to this date will be deleted. You can use a ITUAM date keyword or the date in yyyyymmdd format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Example:</strong> dateToRetainFiles=&quot;PREMON&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This example specifies that all files that were created prior to the previous month will be deleted.</td>
</tr>
<tr>
<td></td>
<td>cleanSubfolders</td>
<td>Optional</td>
<td>Specifies whether the files that are contained in subfolders are deleted. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;true&quot; (the files are deleted)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;false&quot; (the files are not deleted)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

**Table 2-8 • Parameter Element Attributes (Continued)**
Valid Parameters by Program Type

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSOLE</td>
<td>useStandardParameters</td>
<td>Optional</td>
<td>Specifies that if the program type is console, the standard parameters required for all conversion scripts are passed on the command line in the following order:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- LogDate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- RetentionFlag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Feed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- OutputFolder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>These parameters are passed before any other parameters defined for the step. For more information about the standard parameters, see page 2-8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;true&quot; (the standard parameters are passed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- &quot;false&quot; (the standard parameters are not passed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the step type is Process, the default value is &quot;false&quot;. If the step type is ConvertToCSR, the default is &quot;true&quot;.</td>
</tr>
</tbody>
</table>

Table 2-8 • Parameter Element Attributes (Continued)
### Installing ITUAM Data Collectors and Setting Up the System

#### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSOLE (continued)</td>
<td>useCommandProcessor</td>
<td>Optional</td>
<td>Specifies whether the <code>Cmd.exe</code> program should be used to execute a console program. If the <code>Cmd.exe</code> program is not used, then the console program is called using APIs. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• &quot;true&quot; (the <code>Cmd.exe</code> program is used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• &quot;false&quot; (the <code>Cmd.exe</code> program is not used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
<tr>
<td>XMLFileName,</td>
<td></td>
<td>Optional</td>
<td>These attributes are used by the Windows DiskCollector. They specify the name of the XML file used by the collector; the name of the collector; and the collector instance, respectively.</td>
</tr>
<tr>
<td>CollectorName, and</td>
<td></td>
<td></td>
<td>For more information about this collector, see Chapter 4.</td>
</tr>
<tr>
<td>CollectorInstance</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2-8 • Parameter Element Attributes (Continued)**
## Installing ITUAM Data Collectors and Setting Up the System

### Creating Job Files

<table>
<thead>
<tr>
<th>Program Name or Type</th>
<th>Attribute</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| CONSOLE or WSF       | scanFile    | Optional             | This attribute is applicable only if the Smart Scan feature is enabled (see page 2-40 for more information about Smart Scan). 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 `process definition folder\feed subfolder\LogDate.txt`. 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: 

````
scanFile="\\Server1\VMware\Server2\MyFile.txt"
```

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 `scanFile=""` (empty string). |
|                       | TimeoutInMinutes | Optional | Specifies a time limit in minutes or fractional minutes for a console application or script to run before it is automatically terminated. If the application or script run time exceeds the time limit, the step fails and a message explaining the termination is included in the job log file. **Example:** 

````
TimeoutInMinutes="1.5"
```

In this example, the time limit is one and half minutes. The default is 0, which specifies that there is no timeout limit. |

### Table 2-8 • 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 parameters for multiple steps in one location as shown in the examples on page 2-29 and page 2-35. There are two types of attributes that you can use in a Default element: pre-defined and user defined. Table 2-9 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-defined attributes. These are the attributes that are pre-defined for ITUAM Administrator.</td>
<td></td>
</tr>
<tr>
<td>LogDate</td>
<td>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.</td>
</tr>
<tr>
<td>RetentionFlag</td>
<td>This attribute is for future use. Valid values are KEEP or DISCARD.</td>
</tr>
</tbody>
</table>

User-defined attributes. You can define additional default values using the following attributes.

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 2-9 • Default Element Attributes
Using ITUAM Integrator

Note • This section assumes that you are familiar with the job file structure described in Creating Job Files beginning on page 2-27.

ITUAM Integrator is a utility that enables you to modify input data provided in a variety of formats (including CSR or CSR+ files). ITUAM Integrator is run from a job file. For an example, refer to the Universal process in the SampleNightly.xml file.

ITUAM Integrator uses the common XML architecture used for all data collection processes in addition to the following elements that are specific to ITUAM Integrator:

- **Input element.** The Input element defines the input files to be processed. There can be only one Input element defined per process and it must precede the Stage elements. However, the Input element can define multiple files.

- **Stage elements.** ITUAM Integrator processes the data in an input file according to the stages that are defined in the job file XML. A Stage element defines a particular data analysis or manipulation process such as adding an identifier or resource to a record, converting an identifier or resource to another value, or renaming identifiers and resources.

  A Stage element is also used produce an output CSR file or CSR+ file.

Each of these elements and their child elements are described in the following sections.

The following attributes are applicable to both Input and Stage elements:

- **active.** Specifies whether the element is to be included in the integration process. Valid values are "true" [the default] or "false".

- **trace.** Specifies whether trace messages generated by the element are written to the job log file. Valid values are "true" or "false" [the default].

- **stopOnStageFailure.** Specifies whether processing should stop if an element fails. Valid values are "true" [the default] or "false".
Input Element

The Input element has only one valid value name, CSRInput, which parses a file.

Example

```xml
<Input name="CSRInput" active="true">
  <Files>
    <File name="%ProcessFolder%\CurrentCSR.txt"/>
    <File name="%ProcessFolder%\MyCSR.txt"/>
  </Files>
</Input>
```

Where the name attribute defines the location of the file to be processed. As shown in this example, you can define multiple files for processing.

Stage Elements

The following are the valid Stage element names. The stages are presented in alphabetical order with the exception of CSROutput and CSRPlusOutput, which are described first for reference purposes.

CSROutput

The CSROutput stage produces a CSR file.

Example

```xml
<Stage name="CSROutput" active="true">
  <Files>
    <File name="csrafter.txt"/>
  </Files>
</Stage>
```

In this example, the CSR file csrafter.txt file is created. The file is placed in the process definition folder defined by the job file.

CSRPlusOutput

The CSRPlusOutput stage produces a CSR+ file.

Example

```xml
<Stage name="CSRPlusOutput" active="true">
  <Files>
    <File name="csrplusafter.txt"/>
  </Files>
</Stage>
```

In this example, the CSR+ file csrplusafter.txt is produced. The file is placed in the process definition folder defined by the job file.
The Aggregator stage aggregates a file based on the identifiers and resources specified. The identifiers are used for aggregation and the resources are summed. Any resources and identifiers not specified are dropped from the record.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```plaintext
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the Aggregator stage appears as follows:

```xml
<Stage name="Aggregator" active="true">
   <Identifiers>
      <Identifier name="User"/>
      <Identifier name="Feed"/>
   </Identifiers>
   <Resources>
      <Resource name="EXEMRCV"/>
   </Resources>
   <Parameters>
      <Parameter defaultAggregation="false"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```plaintext
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXEMRCV,2
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXEMRCV,2
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",1,EXEMRCV,1
```

The records were aggregated by the identifier values for **User** and **Feed**. Because the resource value **EXBYRCV** in the input file was not defined, it was dropped from the output records.

The sort order is determined by the order in which the identifiers are defined. Precedence is established in sequential order from the first identifier defined to the last. In the preceding example, the identifier **User** is defined first.
Considerations for Using Aggregator

Consider the following when using the Aggregator stage:

- The parameter *defaultAggregation* specifies whether the fields in the record header (start date, end date, account code, etc.) are used for aggregation. The default for *defaultAggregation* is "true".

- The Aggregator stage is memory dependent. The amount of memory affects the amount of time it takes to perform aggregation.

- If an identifier that is defined for aggregation appears in a record with a blank value, it will be included in the aggregated record with a blank value.

Using the Aggregation stage shown on page 2-92, the following is an example aggregation for this scenario:

**Original CSR Records**

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,"",User,"Joe",2,EXEMRCV,1,EXBYRCV,3941

Example,20060117,20060117,00:00:00,23:59:59,,1,User,"Joe",2,EXEMRCV,1,EXBYRCV,2817

**Aggregated CSR Record**

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"",User,"Joe",1,EXEMRCV,2
CreateIdentifierFromIdentifiers

The CreateIdentifierFromIdentifiers stage creates a new identifier for which the initial value is built using one or more current identifier values or substrings within those values.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>20060117,00:00:00,23:59:59,,2</td>
</tr>
<tr>
<td>20060117,00:00:00,23:59:59,,2</td>
</tr>
<tr>
<td>20060117,00:00:00,23:59:59,,2</td>
</tr>
<tr>
<td>20060117,00:00:00,23:59:59,,2</td>
</tr>
<tr>
<td>20060117,00:00:00,23:59:59,,2</td>
</tr>
</tbody>
</table>

If the CreateIdentifierFromIdentifiers stage appears as follows

```xml
<Stage name="CreateIdentifierFromIdentifiers" active="true">
  <Identifiers>
    <Identifier name="Account_Code">
      <FromIdentifiers>
        <FromIdentifier name="User" offset="1" length="5" delimiter="a"/>
        <FromIdentifier name="Feed" offset="1" length="6" delimiter="b"/>
      </FromIdentifiers>
    </Identifier>
  </Identifiers>
  <Parameters>
    <Parameter keepLength="false"/>
    <Parameter modifyIfExists="true"/>
  </Parameters>
</Stage>
```

The output CSR file appears as follows:

```plaintext
Example,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Account_Code,"joebSrvr1a",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",Account_Code,"marybSrvr2a",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Account_Code,"joanbSrvr3a",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Account_Code,"joanbSrvr3a",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Account_Code,"joebSrvr1a",2,EXEMRCV,1,EXBYRCV,2817
```

The identifier **Account_Code** was added. The value for the **Account_Code** identifier is built from the values for the **User** and **Feed** identifiers in the record as defined by the **FromIdentifier** elements. The optional delimiter attribute appends a specified delimiter to the end of the identifier value specified by **FromIdentifier**. In this example, the letter **a** was added to the end of the **FromIdentifier User** identifier value and the letter **b** was added to the end of the **FromIdentifier Feed** identifier value.
Considerations for Using CreateIdentifierFromIdentifiers

Consider the following when using the CreateIdentifierFromIdentifiers stage:

- The FromIdentifier attributes offset and length specify the offset and length of the identifier value to be used. If you want to use the full value for an identifier, use an offset of 1 and the length of the longest identifier value as defined in the input file. For example, if the longest identifier value is 7 characters, type 1 as the offset and 7 as the length.

If you want to use a portion of the identifier value, use the offset position at which you want to start the value and the corresponding length. For example, if you want to start the 7-character identifier value at the second character, type 2 as the offset and 6 as the length. You could also select a shorter length if you wanted to further reduce the identifier value length.

- The parameter keepLength specifies whether the entire length should be included if the length specified is longer than the identifier value. In this case, the value is padded with spaces to meet the maximum length. The default for keepLength is "false".

For example, the values for the FromIdentifiers shown in the XML example on page 2-94, are one character longer than the User and Feed identifier values in the input record. If keepLength was set to "true", the output records would appear as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Sr1 ajoe b",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Sr2 amary b",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Sr3 ajoan b",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Sr3 ajoe b",2,EXEMRCV,1,EXBYRCV,2817

- The order of the FromIdentifier elements defines the order of concatenated values that appear in the new identifier value. In the example on page 2-94, the first FromIdentifier is User, so the value appears before the Feed value.

- If the modifyIfExists parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If modifyIfExists="false" (the default) existing identifier value is not changed.
**CreateIdentifierFromTable**

The `CreateIdentifierFromTable` stage creates a new identifier for which the initial value is built using another identifier’s value as a lookup to a conversion table.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `CreateIdentifierFromTable` stage appears as follows

```
<Stage name="CreateIdentifierFromTable" active="true">
  <Identifiers>
    <Identifier name="Account_Code">
      <FromIdentifiers>
        <FromIdentifier name="User" offset="1" length="4"/>
      </FromIdentifiers>
    </Identifier>
  </Identifiers>
  <Files>
    <File name="Table.txt" type="table"/>
    <File name="Exception.txt" type="exception" format="CSROutput"/>
  </Files>
  <Parameters>
    <Parameter exceptionProcess="true"/>
    <Parameter sort="true"/>
    <Parameter upperCase="false"/>
    <Parameter writeNoMatch="false"/>
    <Parameter modifyIfExists="true"/>
  </Parameters>
</Stage>
```

And the conversion table `Table.txt` appears as follows:

```
joe,,ATM
joan,mary,CCX
```

The output CSR file appears as follows:

```
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Account_Code,"ATM",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Account_Code,"ATM",2,EXEMRCV,1,EXBYRCV,2817
```

The identifier `Account_Code` was added. The value for the `Account_Code` identifier is built from the values defined in the conversion table `Table.txt`. 
Considerations for Using CreateIdentifierFromTable

Consider the following when using the CreateIdentifierFromTable stage:

- Only one new identifier can be specified in the stage.
- If a match is not found in the conversion table and the parameter exceptionProcess is set to "false" (the default), the identifier will be added to the record with a blank value.

  If a match is not found in the conversion table and the parameter exceptionProcess is set to "true", the record will be written to the exception file. The exception file can be in any output format that is supported by ITUAM Integrator. The format is defined by the stage name of the output type. For example, if the stage CSROutput or CSRPlusOutput are active, the exception file is produced as CSR or CSR+ file, respectively.

- If the identifier defined in the FromIdentifier element is not found in the record, the new identifier will be written to the record with a blank value.
- If the parameter sort is set to "true" (the default and recommended), an internal sort of the conversion table is performed.
- The conversion table is case-sensitive. For convenience, you can enter uppercase values in the table and then set the parameter upperCase="true". This ensures that identifier values that are lowercase or mixed case are processed. The default for upperCase is false "false".
- If the parameter writeNoMatch is set to "true", a message is written for the first 1,000 records that do not match an entry in the conversion table. The default for writeNoMatch is "false".

- If the modifyIfExists parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If modifyIfExists="false" (the default) existing identifier value is not changed.

Conversion table rules:

- You can include a default identifier as the last entry in the conversion table by leaving the low and high identifier values empty (e.g., ",,DEFAULTIDENT"). In this case, all records that contain identifier values that do not match an entry in the conversion table will be matched to the default value.

  **Note** • If you have the parameter exceptionProcess set to "true", do not use a default identifier entry. Records will not be written to the exception file.

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

The CreateIdentifierFromValue stage creates a new identifier for which the initial value is specified.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example, 20060117, 20060117, 00:00:00, 23:59:59, ,, 2, Feed, Srvr1, User, "joe", 2, EXEMRCV, 1, EXBYRCV, 3941
Example, 20060117, 20060117, 00:00:00, 23:59:59, ,, 2, Feed, Srvr2, User, "mary", 2, EXEMRCV, 1, EXBYRCV, 3863
Example, 20060117, 20060117, 00:00:00, 23:59:59, ,, 2, Feed, Srvr3, User, "joan", 2, EXEMRCV, 1, EXBYRCV, 2748
Example, 20060117, 20060117, 00:00:00, 23:59:59, ,, 2, Feed, Srvr3, User, "joan", 2, EXEMRCV, 1, EXBYRCV, 3013
Example, 20060117, 20060117, 00:00:00, 23:59:59, ,, 2, Feed, Srvr1, User, "joe", 2, EXEMRCV, 1, EXBYRCV, 3817

If the CreateIdentifierFromValue stage appears as follows:

```xml
<Stage name="CreateIdentifierFromValue" active="true">
  <Identifiers>
    <Identifier name="Break_Room" value="North"/>
  </Identifiers>
  <Parameters>
    <Parameter modifyIfExists="true"/>
  </Parameters>
</Stage>
```

The output CSR file appears as follows:

Example, 20061117, 20061117, 00:00:00, 23:59:59, 1, 3, Feed, "Srvr1", User, "joe", Break_Room, "North", 2, EXEMRCV, 1, EXBYRCV, 3941
Example, 20061117, 20061117, 00:00:00, 23:59:59, 1, 3, Feed, "Srvr2", User, "mary", Break_Room, "North", 2, EXEMRCV, 1, EXBYRCV, 3863
Example, 20061117, 20061117, 00:00:00, 23:59:59, 1, 3, Feed, "Srvr3", User, "joan", Break_Room, "North", 2, EXEMRCV, 1, EXBYRCV, 2748
Example, 20061117, 20061117, 00:00:00, 23:59:59, 1, 3, Feed, "Srvr3", User, "joan", Break_Room, "North", 2, EXEMRCV, 1, EXBYRCV, 3013
Example, 20061117, 20061117, 00:00:00, 23:59:59, 1, 3, Feed, "Srvr1", User, "joe", Break_Room, "North", 2, EXEMRCV, 1, EXBYRCV, 3817

The identifier Break_Room was added with a value of North.

Considerations for Using CreateIdentifierFromValue

Consider the following when using the CreateIdentifierFromValue stage:

- If the modifyIfExists parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If modifyIfExists="false" (the default) existing identifier value is not changed.
**CreateIdentifierFromRegEx**

The `CreateIdentifierFromRegEx` stage creates a new identifier for which the initial value is built using a regular expression to parse the value of a current identifier.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

<table>
<thead>
<tr>
<th>Example</th>
<th>Date</th>
<th>Date</th>
<th>Time</th>
<th>Time</th>
<th>Value</th>
<th>EMAILID</th>
<th>FirstName</th>
<th>LastName</th>
<th>FullName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joe.allen@xyz.com">joe.allen@xyz.com</a>&quot;,2,EXEMRCV,1,EXBYRCV,3941</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:mary.kay@xyz.com">mary.kay@xyz.com</a>&quot;,2,EXEMRCV,1,EXBYRCV,3863</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joan.jet@xyz.com">joan.jet@xyz.com</a>&quot;,2,EXEMRCV,1,EXBYRCV,2748</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joan.jet@xyz.com">joan.jet@xyz.com</a>&quot;,2,EXEMRCV,1,EXBYRCV,3013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joe.allen@xyz.com">joe.allen@xyz.com</a>&quot;,2,EXEMRCV,1,EXBYRCV,2817</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the `CreateIdentifierFromRegEx` stage appears as follows:

```xml
<Stage name="CreateIdentifierFromRegEx" active="true" trace="false" >
  <Identifiers>
    <Identifier name="FirstName">
      <FromIdentifiers>
        <FromIdentifier name="EmailID" regEx="(\w+).(\w+).(\w+)" value="\$1"/>
      </FromIdentifiers>
    </Identifier>
    <Identifier name="LastName">
      <FromIdentifiers>
        <FromIdentifier name="EmailID" regEx="(\w+).(\w+).(\w+)" value="\$2"/>
      </FromIdentifiers>
    </Identifier>
    <Identifier name="FullName">
      <FromIdentifiers>
        <FromIdentifier name="EmailID" regEx="(\w+).(\w+).(\w+)" value="\$2 \$1"/>
      </FromIdentifiers>
    </Identifier>
  </Identifiers>
  <Parameters>
    <Parameter modifyIfExists="true"/>
  </Parameters>
</Stage>
```

The output CSR file appears as follows:

<table>
<thead>
<tr>
<th>Example</th>
<th>Date</th>
<th>Date</th>
<th>Time</th>
<th>Time</th>
<th>Value</th>
<th>EMAILID</th>
<th>FirstName</th>
<th>LastName</th>
<th>FullName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joe.allen@xyz.com">joe.allen@xyz.com</a>&quot;,FirstName,&quot;joe&quot;,LastName,&quot;allen&quot;,FullName,&quot;allen, joe&quot;,2,EXEMRCV,1,EXBYRCV,3941</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:mary.kay@xyz.com">mary.kay@xyz.com</a>&quot;,FirstName,&quot;mary&quot;,LastName,&quot;kay&quot;, FullName,&quot;kay, mary&quot;,2,EXEMRCV,1,EXBYRCV,3863</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joan.jet@xyz.com">joan.jet@xyz.com</a>&quot;,FirstName,&quot;joan&quot;, LastName,&quot;jet&quot;, FullName,&quot;jet, joan&quot;,2,EXEMRCV,1,EXBYRCV,2748</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joan.jet@xyz.com">joan.jet@xyz.com</a>&quot;,FirstName,&quot;joan&quot;, LastName,&quot;jet&quot;, FullName,&quot;jet, joan&quot;,2,EXEMRCV,1,EXBYRCV,3013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>20060117</td>
<td>20060117</td>
<td>00:00:00</td>
<td>23:59:59</td>
<td>,1,EmailID</td>
<td>&quot;<a href="mailto:joe.allen@xyz.com">joe.allen@xyz.com</a>&quot;,FirstName,&quot;joe&quot;, LastName,&quot;allen&quot;, FullName,&quot;allen, joe&quot;,2,EXEMRCV,1,EXBYRCV,2817</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The identifiers `FirstName`, `LastName`, and `FullName` were added.
Considerations for Using CreateIdentifierFromRegEx

Consider the following when using the CreateIdentifierFromRegEx stage:

- The `FromIdentifier` attribute `regEx` defines the regular expression used to parse the identifier value.

- If the `modifyIfExists` parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If `modifyIfExists="false"` (the default) existing identifier value is not changed.
CreateResourceFromConversion

The CreateResourceFromConversion stage creates a new resource for which the initial value is built from other resource values.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,.2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,.2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,.2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,.2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,.2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the CreateResourceFromConversion stage appears as follows:

```xml
<Stage name="CreateResourceFromConversion" active="true">
  <Resources>
    <Resource name="Total">
      <FromResources>
        <FromResource name="EXEMRCV" symbol="a"/>
        <FromResource name="EXBYRCV" symbol="b"/>
      </FromResources>
    </Resource>
    <Parameters>
      <Parameter formula="(a+b)/60"/>
      <Parameter modifyIfExists="true"/>
    </Parameters>
  </Resources>
</Stage>
```

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Total Resource,65.70000
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Total Resource,64.40000
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Total Resource,45.81667
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Total Resource,50.23333
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Total Resource,46.96667

The resource Total Resource was added. The value for the new resource is built from the sum of the existing resource values divided by 60.

Considerations for Using CreateResourceFromConversion

Consider the following when using the CreateResourceFromConversion stage:

- The attribute symbol is restricted to one lowercase letter (a–z).
- The parameter formula is any arithmetic expression using the symbols defined by the symbol attributes.
- If the modifyIfExists parameter is set to "true" and the resource already exists, the existing resource value is modified with the specified value. If modifyIfExists="false" (the default) existing resource value is not changed.
- You will need to add the resource to the CIMSRate table if it does not already exist in the table.
Creating Resources from Values

CreateResourceFromValue

The CreateResourceFromValue stage creates a new resource for which the initial value is specified.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the CreateResourceFromValue stage appears as follows:

<Stage name="CreateResourceFromValue" active="true">
  <Resources>
    <Resource name="Num_Recs" value="1"/>
  </Resources>
  <Parameters>
    <Parameter modifyIfExists="true"/>
  </Parameters>
</Stage>

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1

The resource Num_Recs was added with a value of 1.

Considerations for Using CreateResourceFromValue

Consider the following when using the CreateResourceFromValue stage:

- If the modifyIfExists parameter is set to "true" and the resource already exists, the existing resource value is modified with the specified value. If modifyIfExists="false" (the default) existing resource value is not changed.

- You will need to add the resource to the CIMSRate table if it does not already exist in the table.
**DropFields**

The DropFields stage drops a specified field or fields from the record. The fields can be identifier or resource fields.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20060117,20060117,00:00:00,23:59:59.,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1.EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59.,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1.EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59.,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1.EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59.,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1.EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59.,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1.EXBYRCV,2817
```

If the DropFields stage appears as follows:

```xml
<Stage name="DropFields" active="true">
  <Fields>
    <Field name="Feed"/>
    <Field name="EXEMRCV"/>
  </Resources>
</Stage>
```

The output CSR file appears as follows:

```
Example,20060117,20060117,00:00:00,23:59:59.,1,1,User,"joe",1.EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59.,1,1,User,"mary",1.EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59.,1,1,User,"joan",1.EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59.,1,1,User,"joan",1.EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59.,1,1,User,"joe",1.EXBYRCV,2817
```

The identifier Feed and the resource EXEMRCV have been dropped from the records.

**Considerations for Using DropFields**

Consider the following when using the DropFields stage:

- The field is retained in the record, but the property `skip` is set to `true` so that the field can be used by other stages. The CSROutput or CSRPlusOutput stage checks the `skip` property to determine if the field should be included.

- If you are using the Aggregator stage, this stage is not needed. Only those identifiers and resources specified for aggregation will be included in the output records.
DropIdentifiers

The DropIdentifiers stage drops a specified identifier from the record. This stage is required if you have identifiers and resources with the same name and want to drop the identifier only. However, it is unlikely (and not recommended) that an identifier and a resource have the same name.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817

If the DropIdentifiers stage appears as follows:

<Stage name="DropIdentifiers" active="true">  
<Fields>  
<Field name="Feed">  
</Resources>  
</Stage>

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,2817

The identifier Feed has been dropped from the records. The resource Feed remains.

Considerations for Using DropIdentifiers

Consider the following when using the DropIdentifiers stage:

- The field is retained in the record, but the property skip is set to true so that the field can be used by other stages. The CSROutput or CSRPlusOutput stage checks the skip property to determine if the field should be included.

- If you are using the Aggregator stage, this stage is not needed. Only those identifiers and resources specified for aggregation will be included in the output records.
DropResources

The DropResources stage drops a specified resource from the record. This stage is required if you have identifiers and resources with the same name and want to drop the resource only. However, it is unlikely (and not recommended) that an identifier and a resource have the same name.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817

If the DropResources stage appears as follows:

```xml
<Stage name="DropResources" active="true">
  <Fields>
    <Field name="Feed"/>
  </Fields>
</Stage>
```

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,2817

The resource Feed and has been dropped from the records. The identifier Feed remains.

Considerations for Using DropResources

Consider the following when using the DropResources stage:

- The field is retained in the record, but the property `skip` is set to `true` so that the field can be used by other stages. The CSROutput or CSRPlusOutput stage checks the `skip` property to determine if the field should be included.

- If you are using the Aggregator stage, this stage is not needed. Only those identifiers and resources specified for aggregation will be included in the output records.
ExcludeRecsByDate

The ExcludeRecsByDate stage includes records based on the header end date. Note that for CSR files, the end date in the record header is the same as the end date in the record.

To specify the date, use one of the following date keywords or a date range:

- **PREDAY**
- **CURDAY**
- **RNDATE**
- **PREMON**
- **CURMON**
- **PREWEK**
- Date range in YYYYMMDD format

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060217,20060217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2817

If the ExcludeRecsByDate stage appears as follows:

```xml
<Stage name="ExcludeRecsByDate" active="true">
  <Parameters>
    <Parameter keyword="**PREMON"/>
  </Parameters>
</Stage>
```

Or

```xml
<Stage name="ExcludeRecsByDate" active="true">
  <Parameters>
    <Parameter fromDate="20060101"/>
    <Parameter fromDate="20060131"/>
  </Parameters>
</Stage>
```

If you run the ExcludeRecsByDate stage in February, the output CSR file appears as follows:

Example,20060217,20060217,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,Feed,1,EXBYRCV,3013
Example,20060217,20060217,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,Feed,1,EXBYRCV,2817

Only those records with end dates in February are included.

Considerations for Using ExcludeRecsByDate

Consider the following when using the ExcludeRecsByDate stage:

- This stage drops the entire record. Once the record is dropped, it can no longer be processed by any other stage.
ExcludeRecsByPresence

The ExcludeRecsByPresence stage drops records based on the existence or non-existence of identifiers and/or resources.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817

If the ExcludeRecsByPresence stage appears as follows:

```
<Stage name="ExcludeRecsByPresence" active="true">
  <Identifiers>
    <Identifier name="Feed" exists="true"/>
  </Identifiers>
  <Resources>
    <Resource name="Num_Recs" exists="false"/>
  </Resources>
</Stage>
```

The output CSR file appears as follows:

```
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
```

The first five records in the input file were dropped because they contain the identifier Feed. The last two records in the input file were dropped because they do not contain the resource Num_Recs.

Considerations for Using ExcludeRecsByPresence

Consider the following when using the ExcludeRecsByPresence stage:

- This stage drops the entire record. Once the record is dropped, it can no longer be processed by any other stage.
- Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is dropped.
**ExcludeRecsByValue**

The **ExcludeRecsByValue** stage drops records based on the identifier and/or resource value. If the comparison is false, the record is dropped. The comparison conditions are:

- **GT** (greater than)
- **GE** (greater than or equal to)
- **EQ** (equal to)
- **LT** (less than)
- **LE** (less than or equal to)

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```plaintext
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the **ExcludeRecsByValue** stage appears as follows:

```xml
<Stage name="ExcludeRecsByValue" active="true">
  <Identifiers>
    <Identifier name="User" cond="EQ" value="joan"/>
  </Identifiers>
  <Resources>
    <Resource name="EXBYRCV" cond="GT" value="3000"/>
  </Resources>
</Stage>
```

The output CSR file appears as follows:

```plaintext
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

All records with the **User** identifier value *joan* or with a **EXBYRCV** resource value less than 3000 were dropped.

**Considerations for Using ExcludeRecsByValue**

Consider the following when using the **ExcludeRecsByValue** stage:

- This stage drops the entire record. Once the record is dropped, it can no longer be processed by any other stage.
- Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is dropped.
- If a field specified for exclusion contains a blank value, the record is not dropped.
**IdentifierConversionFromTable**

The **IdentifierConversionFromTable** stage converts an identifier’s value using the identifier’s own value or another identifier’s value as a lookup to a conversion table.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the **IdentifierConversionFromTable** stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
  <Identifiers>
    <Identifier name="Feed">
      <FromIdentifiers>
        <FromIdentifier name="User" offset="1" length="4"/>
      </FromIdentifiers>
    </Identifier>
  </Identifiers>
  <Files>
    <File name="Table.txt" type="table"/>
    <File name="Exception.txt" type="exception" format="CSROutput"/>
  </Files>
  <Parameters>
    <Parameter exceptionProcess="true"/>
    <Parameter sort="true"/>
    <Parameter upperCase="false"/>
    <Parameter writeNoMatch="false"/>
  </Parameters>
</Stage>
```

And the conversion table **Table.txt** appears as follows:

```
joan,,ServerJoan
joe,,ServerJoe
mary,,ServerMary
```

The output CSR file appears as follows:

```
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"ServerJoe",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"ServerMary",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"ServerJoe",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

The value for the **User** identifier was used to determine the new value for the **Feed** identifier as defined in the conversion table **Table.txt**.
Considerations for Using IdentifierConversionFromTable
Consider the following when using the IdentifierConversionFromTable stage:

- If the identifier defined for conversion is not found in the input record, the record is treated as an exception record.

- All other considerations are the same as those for CreateIdentifierFromTable (see page 2-96).
IncludeRecsByDate

The IncludeRecsByDate stage includes records based on the header end date. Note that for CSR files, the end date in the record header is the same as the end date in the record.

To specify the date, use one of the following date keywords or a date range:

- **PREDAY**
- **CURDAY**
- **RNDATE**
- **PREMON**
- **CURMON**
- **PREWEK**
- Date range in YYYYMMDD format

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060217,20060217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060217,20060217,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the IncludeRecsByDate stage appears as follows:

```xml
<Stage name="IncludeRecsByDate" active="true">
  <Parameters>
    <Parameter keyword="**PREMON"/>
  </Parameters>
</Stage>
```

Or

```xml
<Stage name="IncludeRecsByDate" active="true">
  <Parameters>
    <Parameter fromDate="20060101"/>
    <Parameter fromDate="20060131"/>
  </Parameters>
</Stage>
```

If you run the IncludeRecsByDate stage in February, the output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748

Only those records with end dates in January are included.

**Considerations for Using IncludeRecsByDate**

Consider the following when using the IncludeRecsByDate stage:

- This stage drops the entire record if it does not meet the include conditions. Once the record is dropped, it can no longer be processed by any other stage.
IncludeRecsByPresence

The IncludeRecsByPresence stage includes records based on the existence or non-existence of identifiers and/or resources.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the IncludeRecsByPresence stage appears as follows:

<Stage name="IncludeRecsByPresence" active="true">
  <Identifiers>
    <Identifier name="Feed" exists="true"/>
  </Identifiers>
  <Resources>
    <Resource name="Num_Recs" exists="false"/>
  </Resources>
</Stage>

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

The first five records in the input file were included because they contain the identifier Feed. The last two records in the input file were included because they do not contain the resource Num_Recs.

Considerations for Using IncludeRecsByPresence

Consider the following when using the IncludeRecsByPresence stage:

- This stage drops the entire record if it does not meet the include conditions. Once the record is dropped, it can no longer be processed by any other stage.
- Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is included.
IncludeRecsByValue

The IncludeRecsByValue stage drops records based on the identifier and/or resource value. If the comparison is true, the record is included.

The comparison conditions are:

- GT (greater than)
- GE (greater than or equal to)
- EQ (equal to)
- LT (less than)
- LE (less than or equal to)

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the IncludeRecsByValue stage appears as follows:

```xml
<Stage name="IncludeRecsByValue" active="true">
  <Identifiers>
    <Identifier name="User" cond="GE" value="joan"/>
  </Identifiers>
  <Resources>
    <Resource name="EXBYRCV" cond="GT" value="3000"/>
  </Resources>
</Stage>
```

The output CSR file appears as follows:

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817

All records with the User identifier value joan or with a EXBYRCV resource value less than 3000 were included.

Considerations for Using IncludeRecsByValue

Consider the following when using the IncludeRecsByValue stage:

- This stage drops the entire record if it does not meet the include conditions. Once the record is dropped, it can no longer be processed by any other stage.
- If a field specified for inclusion contains a blank value, the record is not included.
- Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is included.
MaxRecords

The MaxRecords stage specifies the number of input records to process. Once this number is reached, processing stops.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the MaxRecords stage appears as follows:

<Stage name="MaxRecords" active="true">
  <Parameters>
    <Parameter number="2"/>
  </Parameters>
</Stage>

The output CSR file appears as follows:

Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863

Only the first two records in the input file were processed.
RenameFields

The RenameFields stage renames specified identifiers and resources.

Example

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srcr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srcr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srcr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srcr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srcr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the RenameFields stage appears as follows:

```xml
<Stage name="RenameFields" active="true">
  <Fields>
    <Field name="User" newName="UserName"/>
    <Field name="EXEMRCV" newName="Emails"/>
    <Field name="EXBYRCV" newName="Bytes"/>
  </Fields>
</Stage>
```

The output CSR file appears as follows:

Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srcr1",UserName,"joe",2,Emails,1,Bytes,3941
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srcr2",UserName,"mary",2,Emails,1,Bytes,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srcr3",UserName,"joan",2,Emails,1,Bytes,2748
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srcr3",UserName,"joan",2,Emails,1,Bytes,3013
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srcr1",UserName,"joe",2,Emails,1,Bytes,2817

Considerations for Using RenameFields

Consider the following when using the RenameFields stage:

- If you rename a resource field, you will need to add the resource to the CIMSRate table if it does not already exist in the table.
**ResourceConversion**

The ResourceConversion stage calculates a resource’s value from the resource’s own value or other resource values.

**Example**

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Server1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Server2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Server3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Server3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Server1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817

If the ResourceConversion stage appears as follows:

```xml
<Stage name="ResourceConversion" active="true">
  <Resources>
    <Resource name="EXEMRCV">
      <FromResources>
        <FromResource name="EXEMRCV" symbol="a"/>
      </FromResources>
    </Resource>
  </Resources>
  <Parameters>
    <Parameter formula="a*60"/>
  </Parameters>
</Stage>
```

The output CSR file appears as follows:

Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Server1",User,"joe",2,EXEMRCV,60,EXBYRCV,3941
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Server2",User,"mary",2,EXEMRCV,60,EXBYRCV,3863
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Server3",User,"joan",2,EXEMRCV,60,EXBYRCV,2748
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Server3",User,"joan",2,EXEMRCV,60,EXBYRCV,3013
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Server1",User,"joe",2,EXEMRCV,60,EXBYRCV,2817

The new value for the resource EXEMRCV is calculated by multiplying the existing value by 60.

**Considerations for Using ResourceConversion**

Consider the following when using the ResourceConversion stage:

- The attribute symbol is restricted to one lowercase letter (a–z).
- The parameter formula is any arithmetic expression using the symbols defined by the symbol attributes.
Sort

The Sort stage sorts records in the output file based on the specified identifier value or values. Records can be sorted in ascending or descending order.

Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20060117,20060117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the Sort stage appears as follows:

```
<Stage name="Sort" active="true">
  <Identifiers>
    <Identifier name="User" length="6"/>
    <Identifier name="Feed" length="7"/>
  </Identifiers>
  <Parameters>
    <Parameter Order="Descending"/>
  </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2718
Example,20061117,20061117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20060117,20060117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
```

The sort order is determined by the order in which the identifiers are defined. Precedence is established in sequential order from the first identifier defined to the last. In the preceding example, the identifier User is defined first.
Considerations for Using Sort

Consider the following when using the Sort stage:

- The parameter length is optional. This specifies the length within the identifier value that you want to use for sorting. If you want to use the entire value, the length parameter is not required.
  
  If the length of the actual identifier value is less than the specified length, blanks are used to pad the length.

- The default sort order is ascending.

- The Sort stage is memory dependent. The amount of memory affects the amount of time it takes to perform aggregation.
Running ITUAM Data Collectors

This section describes how to use ITUAM Job Runner to execute the data collection process. You should determine the frequency that you want to run ITUAM Data Collectors as described in Data Processing Frequency before you run ITUAM 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 and applications/databases is 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 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 ITUAM 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.
Installing ITUAM Data Collectors and Setting Up the System

Running ITUAM Data Collectors

Running ITUAM Job Runner

You can run ITUAM 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 ITUAM Job Runner from Task Scheduler:**

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 ITUAM Job Runner in the **Run** box as shown in the following example:

   "C:\Program Files\IBM\ITUAM\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 2-121.
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** • It is recommended that you create one job file for data collection. If you have multiple job files, you must schedule a separate instance of ITUAM Job Runner to run each file. If the job files run concurrently, ITUAM will attempt to load the output files from each job file into the database simultaneously, which might result in errors.
To run ITUAM Job Runner from the command prompt:

At the C:\Program Files\IBM\ITUAM\Process Engine> prompt type cimsjobrunner.exe followed by the optional parameters described in Optional Parameters. Or from any prompt, type "C:\Program Files\IBM\ITUAM\Process Engine\CIMSJobRunner.exe" followed by the optional parameters.

Examples

C:\Program Files\IBM\ITUAM\Process Engine>CIMSJobRunner.exe Nightly.xml

Or

C:\>"C:\Program Files\IBM\ITUAM\Process Engine\CIMSJobRunner.exe" Nightly.xml

In the preceding examples, Nightly.xml is the job file name.

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

In addition to the required parameter for the job file name, you can supply the following optional parameters for ITUAM Job Runner:

<job id> <process id> <step id> <date literal | keyword>

Where:

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 ITUAM 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 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 2-28.
## Running ITUAM Data Collectors

### Examples

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CimsJobRunner.exe Nightly.xml</td>
<td>ITUAM Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml.</td>
</tr>
<tr>
<td>CimsJobRunner.exe Nightly.xml Nightly</td>
<td>ITUAM 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.</td>
</tr>
<tr>
<td>CimsJobRunner.exe Nightly.xml Nightly All DatabaseLoad</td>
<td>ITUAM Job Runner runs only the active DatabaseLoad steps in all processes in the Nightly job. No other steps in the job file are run.</td>
</tr>
<tr>
<td>CimsJobRunner.exe Nightly.xml All All All 20060604</td>
<td>ITUAM Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the LogDate parameter 20060604.</td>
</tr>
<tr>
<td>CimsJobRunner.exe Nightly.xml All All All RNDATE</td>
<td>ITUAM Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the LogDate parameter RNDATE.</td>
</tr>
</tbody>
</table>
Operating System Data Collectors

This chapter contains instructions for setting up and running ITUAM Data Collectors for operating systems. You should have a good understanding of the ITUAM Data Collector system architecture as described in the ITUAM Data Collectors Architecture section beginning on page 2-4 before continuing with the collector-specific information in this chapter.

Windows Process Data Collector .................................................. 3-2
Creating a Log On User Account for the Windows Process Collector Service (Optional) ........ 3-2
System Configuration Options for the Windows Process Collector ................................. 3-4
Installing the Windows Process Collector ........................................ 3-9
Windows Process Collector Log File Format ....................................... 3-21
Identifiers and Resources Collected From the Windows Process Collector Log File .......... 3-25
Setting Up the Windows Process Collector ........................................ 3-26
Running the Windows Process Collector .......................................... 3-32

VMware Data Collector ................................................................. 3-33
Identifiers and Resources Collected by the VMware Collector ................................. 3-33
Setting Up the VMware Collector .................................................. 3-34
Running the VMware Collector ..................................................... 3-36
Windows Process Data Collector

The 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 Windows Process Collector Log File Format on page 3-21). 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 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 Windows Process Collector Service (Optional)

The Windows Process collector includes a Windows service that supports the collector. The service name is ITUAM Process Collector. By default, the service runs under the Local System user account. It is recommended 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:

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 Polices > 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**

**Note** • If you install the Windows Process collector on multiple servers, the service is installed on each server and you must repeat the following steps on each server.

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

1 From Control Panel, navigate to Administrative Tools > Local Security Policy > Local Polices > 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 Windows Process collector log file is written.

If the Windows Process collector is currently running under Local System, modify the service to run under the appropriate account.
System Configuration Options for the Windows Process Collector

You can use any of the following system configurations to process the log files produced by the 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 Windows Process collector and then pulled to the central ITUAM Data Collectors server for processing.

![System Configuration 1 Diagram](image)

**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 3-26.
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 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 ITUAM Data Collectors server. The log files on the central server are then processed into CSR files.

**Note** • The ITUAM installation does not include a script for transferring files. If you need assistance developing a script, contact IBM. 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 Windows Process Administrator GUI (see page 3-20).

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 3-26.
Using Environment Variables With the File Transfer Script

The 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 Windows Process collector will expand the environment variables before launching the script.

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 ITUAM 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.

For an example of the job file XML that supports this configuration, see *Job File Example for Configurations 1, 2, and 3* on page 3-26.
Configuration 4: Generating CSR Files on the Collector Server

Note • This configuration is usually not recommended. For more information, contact IBM.

In this configuration, the log files are written to a log folder on the server running the 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 ITUAM Data Collectors server; however, the CSR files must be processed by ITUAM Processing Engine on the central server.

For an example of the job file XML that supports this configuration, see Job File Examples for Configuration 4 on page 3-29.
Installing the Windows Process Collector

To use the Windows Process collector, you must have the collector installed on the central ITUAM Data Collectors server as described on page 2-3.

In addition to installation on the central server, you must install the 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.)

You can install the Windows Process collector in either of the following ways:

- **Remote installation.** This method enables you to automatically deploy the Windows Process collector to multiple servers.
- **Manual installation.** This method requires that you manually install the collectors on each server. This method requires more steps to prepare for and perform the installation.

These methods are described separately in the following sections.

**Note** • Installation on servers other than the central ITUAM Data Collectors server does not include ITUAM Processing Engine, which processes the CSR files created by ITUAM Aggregation Engine and loads the output data into the database. To process CSR files, you need to process the files on the central server.

Installing Remotely

The following are the steps required to remotely install the Windows Process collector.

**Modify the Sample Deployment Job File**

A sample job file for deploying the Windows Process collector remotely is provided in the ...

IBM\ITUAM\JobFiles folder. This file is named SampleDeployProcessCollector.xml and can be modified for your organization.

**Note** • Rename the modified job file so that it is not overwritten when you upgrade to a new version of ITUAM Data Collectors.

The contents of the SampleDeployProcessCollector.xml job file are shown here. Note that this file contains only one deployment step. A separate deployment step is required for each server that you want to install the Windows Process collector on. To deploy to multiple servers, simply copy the deployment step (that is, copy everything from the opening to the closing Step tag) for each server and modify the values in the step as needed.

You can also use the SampleDeployProcessCollector.xml job file to remove the collector from a server.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Jobs xmlns="http://www.cimslab.com/CIMSJobs.xsd">
  <Job id="DeployProcessCollector" description="Deploy the Process Collector Agent" active="true" dataSourceId=""/>
</Jobs>
```
Operating System Data Collectors

Windows Process Data Collector

```xml
<Process id="DeployProcessCollector"
    description="Deployment of the Process Collector Agent"
    joblogShowStepOutput="true"
    joblogShowStepParameters="true"
    active="true">
    <Steps stopOnStepFailure="true">
        <Step id="Server1 Deployment"
            description="Server1 Deployment"
            type="ConvertToCSR"
            programName="rpd \rpd.jar"
            programType="java"
            active="true">
            <Parameters>
                <Parameter Action="-install"/>
                <Parameter Host="-host test-install"/>
                <Parameter UserId="-userid rxa"/>
                <Parameter Password="-password rxa"/>
                <!--Parameter KeyFilename="-keyfilename yourkeyfilename"-->
                <!--Parameter Manifest="-manifest DeploymentManifest"-->
                <!--Parameter Protocol="-protocol win | ssh"-->
                <!--Parameter JavaCommandLine='-cp "%CIMSInstallLocation%\scripts\rpd\rpd.jar;%CIMSInstallLocation%\scripts\rpdTivoliCommonCLI.jar;%CIMSInstallLocation%\scripts\rpd\jlanclient.jar;%CIMSInstallLocation%\scripts\rpd\jlog.jar;%CIMSInstallLocation%\scripts\rpd\jlogEnglish.jar;%CIMSInstallLocation%\scripts\rpdvremoteaccess.jar;%CIMSInstallLocation%\scripts\rpdvssh.jar" com.ibm.tivoli.ituam.rpd.RemoteProductDeployment'-->
                <Parameter Verbose="-verbose"/>
                <Parameter SourcePath='"-sourcepath %CIMSInstallLocation%\Collectors\CIMSWinProcess"'/>
            </Parameters>
        </Step>
    </Steps>
</Process>

<Process id="RemoveProcessCollector"
    description="Removal of the Process Collector Agent"
    joblogShowStepOutput="true"
    joblogShowStepParameters="true"
    active="false">
    <Steps stopOnStepFailure="true">
        <Step id="Server1 Removal"
            description="Server1 Removal"
            type="ConvertToCSR"
            programName="rpd \rpd.jar"
            programType="java"
            active="true">
            <Parameters>
                <!--Parameter KeyFilename="-keyfilename yourkeyfilename"-->
                <!--Parameter Manifest="-manifest DeploymentManifest"-->
                <!--Parameter Protocol="-protocol win | ssh"-->
                <!--Parameter JavaCommandLine='-cp "%CIMSInstallLocation%\scripts\rpd\rpd.jar;%CIMSInstallLocation%\scripts\rpdTivoliCommonCLI.jar;%CIMSInstallLocation%\scripts\rpd\jlanclient.jar;%CIMSInstallLocation%\scripts\rpd\jlog.jar;%CIMSInstallLocation%\scripts\rpd\jlogEnglish.jar;%CIMSInstallLocation%\scripts\rpdvremoteaccess.jar;%CIMSInstallLocation%\scripts\rpdvssh.jar" com.ibm.tivoli.ituam.rpd.RemoteProductDeployment'-->
                <Parameter Verbose="-verbose"/>
                <Parameter SourcePath='"-sourcepath %CIMSInstallLocation%\Collectors\CIMSWinProcess"'/>
            </Parameters>
        </Step>
    </Steps>
</Process>
```
<Parameters>
  <Parameter JavaCommandLine = '-cp "%CIMSInstallLocation%\scripts\rpdp\rpdp.jar;%CIMSInstallLocation%\scripts\rpdp\TivoliCommonCLI.jar;%CIMSInstallLocation%\scripts\rpdp\jlanclient.jar;%CIMSInstallLocation%\scripts\rpdp\jlog.jar;%CIMSInstallLocation%\scripts\rpdp\jlogEnglish.jar;%CIMSInstallLocation%\scripts\rpdp\remoteaccess.jar;%CIMSInstallLocation%\scripts\rpdp\ssh.jar" com.ibm.tivoli.ituam.rpd.RemoteProductDeployment'/>  
  <Parameter Action = "-remove"/>
  <Parameter Host = "-host test-install"/>
  <Parameter UserId = "-userid rxa"/>
  <Parameter Password = "-password rxa"/>
  <!--Parameter KeyFilename = "-keyfilename yourkeyfilename"/-->  
  <!--Parameter Manifest = "-manifest DeploymentManifest"/-->  
  <!--Parameter Protocol = "-protocol win | ssh"/-->  
  <Parameter RPDParameters = '-parameters AccountingInterval=86400; AccountingIntervalCommand=""; AccountingIntervalTime="00:00"; LogFileExtension=".txt"; LogFilePath="%ITUAMInstallPathRemote%CIMSWinProcessLogs"; LogFilePrefix="CIMSProcessLog-"; SamplingInterval=1; UseAccountingIntervalTime=No; UseLocalTime=Yes; WriteIntervalEndRecords=No;'/>
  <Parameter Verbose = "-verbose"/>
  <Parameter SourcePath = '-sourcepath "%CIMSInstallLocation%\Collectors\CIMSWinProcess"'/>
</Parameters>
</Step>
</Steps>
</Process>
</Job>
</Jobs>

**SampleDeployProcessCollector.xml Job File Structure**

The *SampleDeployProcessCollector.xml* job file follows the same structure as described in *Job File Structure* on page 2-44.

Table 3-1 describes the parameters that are specific to this job file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaCommandLine</td>
<td>Required</td>
<td>Do not change this parameter.</td>
</tr>
<tr>
<td>ParameterAction</td>
<td>Required</td>
<td>Do not change this parameter.</td>
</tr>
<tr>
<td>Host</td>
<td>Required</td>
<td>The IP address or DNS name of the server that you want to install the Windows Process collector on.</td>
</tr>
</tbody>
</table>

**Table 3-1 • SampleDeployProcessCollector.xml Job File Parameters**
### Operating System Data Collectors

#### Windows Process Data Collector

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserId Password</td>
<td>Required</td>
<td>The user account and password for the host server. The user account must belong to the Administrators group.</td>
</tr>
<tr>
<td>KeyFilename</td>
<td>Optional</td>
<td>If you are using the ssh (Secure Shell) protocol (see the Protocol parameter on page 3-12), the SSH server’s host key.</td>
</tr>
<tr>
<td>Manifest</td>
<td>Optional</td>
<td>The path to the DeploymentManifest.xml file. This parameter is required only if the file is in a location other than the ...IBM\ITUAM\Collectors\CIMSWinProcess folder on the ITUAM application server. The DeploymentManifest.xml file defines the default parameters for the collectors. You can use these default parameters specified in the manifest file or you can override the default values using the job file parameter RPDParameters (see page 3-13). It is recommended that you do not change the default values in the manifest file.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Optional</td>
<td>The protocol used to deploy the installation files: win (Windows) or ssh (Secure Shell)</td>
</tr>
</tbody>
</table>

Table 3-1  *SampleDeployProcessCollector.xmlJob File Parameters (Continued)*
RPDParameters | Optional | These are the same parameters that are in the ...Collectors\CIMSWinProcess\DeploymentManifest.xml file. These parameters are used to configure the log file produced by the Windows Process collector. The parameter values that you define here override the default values in the manifest file.

The following is a brief description of each of these parameters. For detailed descriptions, see Enabling Windows Process Logging on page 3-18.

- **AccountingInterval**—If this parameter is set to a positive number (in seconds), this parameter creates interval records in the log file at the specified number of seconds. If you do not want to create interval records, set this parameter to a negative number. The default is "86400".

- **AccountingIntervalCommand**—This parameter enables you to enter a command that will be executed at each logging interval. The default is blank.

- **AccountingIntervalTime**—This parameter enables you to set a time each day (in 24 hour format) to produce interval records. The default is "00:00".

To use this parameter, the UseAccountingIntervalTime parameter must be set to "yes".

- **UseAccountingIntervalTime**—If this parameter is set to "yes", the AccountingIntervalTime parameter value is used to produce interval records.

If this parameter is set to "no" (the default), the AccountingInterval parameter value is used to produce interval records.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDParameters</td>
<td>Optional</td>
<td>These are the same parameters that are in the ...Collectors\CIMSWinProcess\DeploymentManifest.xml file. These parameters are used to configure the log file produced by the Windows Process collector. The parameter values that you define here override the default values in the manifest file.</td>
</tr>
</tbody>
</table>

The following is a brief description of each of these parameters. For detailed descriptions, see Enabling Windows Process Logging on page 3-18.

- **AccountingInterval**—If this parameter is set to a positive number (in seconds), this parameter creates interval records in the log file at the specified number of seconds. If you do not want to create interval records, set this parameter to a negative number. The default is "86400".

- **AccountingIntervalCommand**—This parameter enables you to enter a command that will be executed at each logging interval. The default is blank.

- **AccountingIntervalTime**—This parameter enables you to set a time each day (in 24 hour format) to produce interval records. The default is "00:00".

To use this parameter, the UseAccountingIntervalTime parameter must be set to "yes".

- **UseAccountingIntervalTime**—If this parameter is set to "yes", the AccountingIntervalTime parameter value is used to produce interval records.

If this parameter is set to "no" (the default), the AccountingInterval parameter value is used to produce interval records.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDParameters</td>
<td>Optional</td>
<td>These are the same parameters that are in the ...Collectors\CIMSWinProcess\DeploymentManifest.xml file. These parameters are used to configure the log file produced by the Windows Process collector. The parameter values that you define here override the default values in the manifest file.</td>
</tr>
</tbody>
</table>

The following is a brief description of each of these parameters. For detailed descriptions, see Enabling Windows Process Logging on page 3-18.

- **AccountingInterval**—If this parameter is set to a positive number (in seconds), this parameter creates interval records in the log file at the specified number of seconds. If you do not want to create interval records, set this parameter to a negative number. The default is "86400".

- **AccountingIntervalCommand**—This parameter enables you to enter a command that will be executed at each logging interval. The default is blank.

- **AccountingIntervalTime**—This parameter enables you to set a time each day (in 24 hour format) to produce interval records. The default is "00:00".

To use this parameter, the UseAccountingIntervalTime parameter must be set to "yes".

- **UseAccountingIntervalTime**—If this parameter is set to "yes", the AccountingIntervalTime parameter value is used to produce interval records.

If this parameter is set to "no" (the default), the AccountingInterval parameter value is used to produce interval records.
### Operating System Data Collectors

**Windows Process Data Collector**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| RPDParameters                     | Optional             | ▪ LogFileExtension—The extension that you want to use for the Windows Process collector log file. The default is ".txt".

▪ LogFilePath—The folder that you want to store the log files. The default is "%ITUAMInstallPathRemote% CIMSWinProcessLogs".

▪ LogFilePrefix—The default name for the log file is CIMSProcessLog-yyyyymmdd.txt. You can use the default prefix "CIMSProcessLog-" or replace with the a prefix of your choice or no prefix.

▪ SamplingInterval—The number of seconds that you want to begin tracking new processes. The default is 1 second ("1").

▪ UseLocalTime—The default "yes" specifies that the local time set for the computer is used in the date and time fields in the log file. If you set this parameter to "no", Universal Time Coordinate (UTC) is used in the log file.

▪ WriteIntervalEndRecords—The default "no" specifies that end records are not included in the log file. If you set this parameter to "yes", end records are included in the log file in addition to start and interval records.

▪ LogFileFieldDelimiter—The default delimiter for the log file records. The default is a tab ("\t").

---

**Table 3-1 • SampleDeployProcessCollector.xmlJob File Parameters (Continued)**
Operating System Data Collectors

Windows Process Data Collector

Run the Deployment Job File

The following example shows how to run the SampleDeployProcessCollector.xml job file from the command prompt. The job file name is passed as a parameter to ITUAM Job Runner (CIMSJobRunner.exe).

C:\Program Files\IBM\ITUAM\Process Engine>CIMSJobRunner.exe SampleDeployProcessCollector.xml

For more information about running job files using ITUAM Job Runner, see Running ITUAM Job Runner on page 2-120.

The ITUAM Process Collector service is installed and started on each of the servers specified in the deployment job file. Service is automatically restarted when the system is restarted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbose</td>
<td>optional</td>
<td>The value -verbose specifies that additional information is included in the job log file for debugging and troubleshooting purposes. If you do not include this parameter or leave the parameter value blank, this additional information is not provided in the log file.</td>
</tr>
<tr>
<td>SourcePath</td>
<td>Required</td>
<td>The path to the ...Collectors\CIMSWinProcess folder. This folder contains the installation file. Do not change this value unless the CIMSWinProcess folder is not in the default location.</td>
</tr>
</tbody>
</table>
Installing Manually
The following are the steps required to manually install the Windows Process Collector.

**Run the CIMSWinProcessSetup.exe Program**

1. Log on to Windows as a user in the Administrators group.
2. Click the Windows Start button, and then click Run.
3. Enter the path to the setup program CIMSWinProcessSetup.exe and then click OK. This program is in the ...IBM\ITUAM\Collectors\CIMSWinProcess folder on the ITUAM application server.

   The setup wizard appears.
4. Click Next.
5. Choose the default location for installation (C:\Program Files\IBM\ITUAM) 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 ITUAM Process Collector—Service check box selected and all other check boxes unselected). The service collector is the most commonly used process collector. Contact IBM before selecting the ITUAM Process Collector—User32 Collector check box.
7. Click Next, and then click Install.
8. If the 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 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 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 IBM for troubleshooting purposes. For more information about this program, contact IBM.
  - The executable program for the Windows Process collector’s administrative program, CIMSWinPServiceAdmin.exe.
  - The conversion script, CIMSWinProcess.wsf. In most cases, this file is used on the central ITUAM 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 Windows Process collector as shown in the configuration on page 3-8.
Enabling Windows Process Logging

The 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 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 ▶ ITUAM ▶ Collectors ▶ ITUAM Windows Process Administrator—Service and set the following options:

**Note** • If you install the Windows Process collector remotely (see Installing Remotely on page 3-9), most parameters in the deployment job file correlate to one of the following options in the GUI. The correlating job file parameter is noted in the GUI description.

- **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 Windows Process collector or on the central ITUAM Data Collectors server, depending on the system configuration that you are using for collection and processing (see System Configuration Options for the 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\IBM\ITUAM\CIMSWinProcessLogs (if you installed the Windows Process collector in the default location). The use of a UNC path for the log file location is recommended.

    **Job File Parameter:** LogFilePath (see page 3-14).

    - **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).

      **Job File Parameter:** LogFilePrefix (see page 3-14).

    - **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, 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.

      **Job File Parameter:** UseLocalTime (see page 3-14).
### 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 began 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 *Windows Process Collector Log File Format* on page 3-21.)

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

*Job File Parameter:* **SamplingInterval** (see page 3-14). Note that this value can be entered only in seconds in the job file.

### 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 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 IBM.

*Job File Parameter:* **WriteIntervalEndRecords** (see page 3-14).
• **Write accounting records every.** This option enables you to create interval records at a set number of seconds, minutes, or hours from when the 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.

*Job File Parameter: AccountingInterval* (see page 3-13). Note that this value can be entered only in seconds in the job file.

• **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.

*Job File Parameter: UseAccountingIntervalTime and AccountingIntervalTime (see page 3-13).*

• **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.

*Job File Parameter: AccountingIntervalCommand (see page 3-13).*

• **Control Service.** Click this button to open the Service Control dialog box to start or stop the ITUAM Process Collector 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.
Windows Process Collector Log File Format

The following table describes the record fields in the log file produced by the 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.

- **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.
**Operating System Data Collectors**

*Windows Process Data Collector*

---

**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 3-19).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Type</td>
<td>S = Start of process (note that this does not appear if interval accounting is not used).</td>
</tr>
<tr>
<td></td>
<td>I = Interval (note that this does not appear if interval accounting is not used).</td>
</tr>
<tr>
<td></td>
<td>E = End of process (this record appears if you do not enable interval accounting or if you enable interval accounting and select the <strong>Write End records</strong> check box).</td>
</tr>
<tr>
<td>ProcessID</td>
<td>Process identifier (PID) assigned to the process by the operating system.</td>
</tr>
<tr>
<td>ParentProcessID</td>
<td>The PID for the entity that created the process. Assigned by the operating system.</td>
</tr>
<tr>
<td>ProcessName</td>
<td>The name of the process.</td>
</tr>
<tr>
<td>ProcessPath</td>
<td>The path where the process executable is located.</td>
</tr>
<tr>
<td>MachineName</td>
<td>The name of the computer running the process.</td>
</tr>
<tr>
<td>UserName</td>
<td>The name of the user that created the process.</td>
</tr>
<tr>
<td>TerminalServicesSessionID</td>
<td>If Microsoft Terminal Services is used to access the process on the computer, the session ID.</td>
</tr>
<tr>
<td>CreateDateTime</td>
<td>The date and time that the process was created.</td>
</tr>
<tr>
<td>ExitDateTime</td>
<td>The date and time that the entire process ended.</td>
</tr>
<tr>
<td>ExitCode</td>
<td>The result from the completion of the process.</td>
</tr>
<tr>
<td>IntervalStartDate</td>
<td>If using interval accounting, the date and time that the interval started.</td>
</tr>
<tr>
<td>IntervalEndDate</td>
<td>If using interval accounting, the date and time that the interval ended.</td>
</tr>
<tr>
<td>ElapsedTimeSecs</td>
<td>The total elapsed time in seconds for the process.</td>
</tr>
<tr>
<td>CPUPTimeSecs</td>
<td>The total elapsed CPU time in seconds for the process. This field is the sum of KernelCPUPTimeSecs and the UserCPUPTimeSecs fields.</td>
</tr>
</tbody>
</table>

**Table 3-2 • Windows Process Collector Log File Format**
### Field Name | Description/Values
---|---
KernelCPUTimeSecs | The total elapsed time in seconds that the process spent in kernel mode. (For a description of kernel mode, see *About Kernel Mode and User Mode* on page 3-24).
UserCPUTimeSecs | The total elapsed time in seconds that the process spent in user mode. (For a description of user mode, see *About Kernel Mode and User Mode* on page 3-24).
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.
SystemProcessorCount | The number of processors on the computer.
EligibleProcessorCount | 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 3-2 • 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 Windows Process Collector Log File

By default, the following fields in the 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.

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

Table 3-3 • Default Windows Process Identifiers and Resources
Setting Up the Windows Process Collector

You will need to set up an XML job file on the central ITUAM 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 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 ITUAM Data Collectors server, set up an XML job file for the Windows Process collector as described in Creating Job Files on page 2-27.

The following is a sample 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).

```xml
<?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="true"
       smtpSendJobLog="true"
       smtpServer="mail.ITUAMCustomerCompany.com"
       smtpFrom="ITUAM@ITUAMCustomerCompany.com"
       smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
       stopOnProcessFailure="false">
    <Process id="CIMSWinProcess"
             description="Process for 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>
      </Steps>
    </Process>
  </Job>
</Jobs>
```
For a description of the Parameter element attributes that are specific to the Windows Process collector (that is, the parameters provided for the `ConvertToCSR` step), see Table 3-4. These parameters are used by the conversion script, `CIMSWinProcess.wsf`.

For a description of all other elements and attributes in the job file, see Job File Structure on page 2-44.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogDate</td>
<td>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 Specifying Log Dates for Collection on page 2-4.</td>
</tr>
<tr>
<td>RetentionFlag</td>
<td>This parameter is for future use.</td>
</tr>
</tbody>
</table>

Table 3-4  ·  CIMSWinProcess.wsf Parameters
### Operating System Data Collectors

#### Windows Process Data Collector

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed</td>
<td>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 CIMSWinProcess.wsf script used to convert the file, you can also use &quot;Self&quot; and the server name is defined automatically (see the example on page 3-29). 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 Feed Subfolder on page 2-14). This is the CSR file that is processed by the Scan program. This parameter is included as an identifier in the CSR file.</td>
</tr>
<tr>
<td>OutputFolder</td>
<td>The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program. By default, the output folder is defined by the Process id attribute in the job file. For example, if the Process id=&quot;CIMSWinProcess&quot;, the output folder is CIMSWinProcess. This parameter is required only if you are running the CIMSWinProcess.wsf 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.</td>
</tr>
<tr>
<td>LogFolder</td>
<td>The location of the log file to be processed.</td>
</tr>
</tbody>
</table>

*Table 3-4 • 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 Windows Process collector or the central ITUAM Data Collectors server.

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

On the computer running the Windows Process collector, set up a job file as described in Creating Job Files on page 2-27. 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 CIMS\WinProcess\Server1 folder on the collector server to the CIMS\WinProcess\Server1 folder on the central server (ITUAM).

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 2-4.)

```xml
<?xml version="1.0" encoding="utf-8"?>
<Jobs xmlns="http://www.cimslab.com/CIMS\Jobs.xsd">
  <Job id="Nightly">
    <description>Daily collection</description>
    <active>true</active>
    <dataSourceId=""/>
    <joblogShowStepParameters>true</joblogShowStepParameters>
    <joblogShowStepOutput>true</joblogShowStepOutput>
    <processPriorityClass>Low</processPriorityClass>
    <joblogWriteToTextFile>true</joblogWriteToTextFile>
    <joblogWriteToXMLFile>true</joblogWriteToXMLFile>
    <smtpSendJobLog>true</smtpSendJobLog>
    <smtpServer>mail.ITUAMCustomerCompany.com</smtpServer>
    <smtpFrom>ITUAM@ITUAMCustomerCompany.com</smtpFrom>
    <smtpTo>John.ITUAMUser@ITUAMCustomerCompany.com</smtpTo>
    <stopOnProcessFailure>false</stopOnProcessFailure>
    <Process id="CIMS\WinProcess">
      <description>Process for Windows Process Collection</description>
      <active>true</active>
    </Process>
    <Steps>
      <Step id="Server1 Collection">
        <description>Server1 CIMS\WinProcess</description>
        <type>ConvertToCSR</type>
        <programName>CIMS\WinProcess\CIMS\WinProcess.wsf</programName>
        <programType>wsf</programType>
        <active>true</active>
      </Step>
      <Parameters>
        <Parameter Feed="Self"/>
        <Parameter LogFolder="\Server1\CIMS\WinProcess\Logs"/>
      </Parameters>
    </Step>
  </Job>
</Jobs>
```
On the central ITUAM 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:

```xml
<Process id="CIMSWinProcess"
    description="Process for 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>
    </Steps>
</Process>
```
To write the CSR files to the central ITUAM Data Collectors server:

On the computer running the Windows Process collector, set up a job file as described in Creating Job Files on page 2-27. 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 ITUAM Data Collectors server (ITUAM) 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="\ITUAM\CIMSWinProcess"/>
      </Parameters>
    </Step>
  </Steps>
</Process>
```

On the central ITUAM 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>
  </Steps>
</Process>
```
Operating System Data Collectors

Windows Process Data Collector

Running the Windows Process Collector

Use ITUAM Job Runner to run the Windows Process collector as described in Running ITUAM Job Runner on page 2-120.
VMware Data Collector

The ITUAM 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 resources are pre-loaded in the CIMSRate table.

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)
- VMCPUUGUA (CPU usage guaranteed)
- VMDSKRED (disk kilobytes read)
- VMDSKWRI (disk kilobytes written)
- VMMEMACT (memory kilobytes active)
- VMMEGMRT (memory kilobytes granted)
- VMNETREC (network kilobytes read)
- VMNETTRN (network kilobytes transferred)
Setting Up the VMware Collector

Create an ITUAM Data Source

You need to create an ITUAM Data Source that points to the VMware VirtualCenter database. For the steps required to create an ITUAM Data Source, refer to the IBM Tivoli Usage and Accounting Manager Administrator's Guide.

Set Up the Job File

On the central ITUAM Data Collectors server, set up an XML job file for the VMware collector as described in Creating Job Files on page 2-27. The following is an example process for the collector in the job file.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Jobs xmlns="http://www.cimslab.com/CIMSJobs.xsd">
  <Job id="DeployProcessCollector"
    description="Deploy the Process Collector Agent"
    active="true"
    dataSourceId="" joblogShowStepParameters="true"
    joblogShowStepOutput="true"
    processPriorityClass="Low"
    joblogWriteToTextFile="true"
    joblogWriteToXMLFile="true"
    smtpSendJobLog="false"
    smtpServer="mail.ITUAMCustomerCompany.com"
    smtpFrom="ITUAM@ITUAMCustomerCompany.com"
    smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
    stopOnProcessFailure="false">
    <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>
    </Steps>
  </Job>
</Jobs>
```
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 3-5. These parameters are used by the conversion script, VMware.wsf.

For a description of all other elements and attributes in the job file, see Job File Structure on page 2-44.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogDate</td>
<td>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 Specifying Log Dates for Collection on page 2-4.</td>
</tr>
<tr>
<td>RetentionFlag</td>
<td>This parameter is for future use.</td>
</tr>
</tbody>
</table>

Table 3-5 • VMware.wsf Parameters
Operating System Data Collectors

VMware Data Collector

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed</td>
<td>The name of the server that contains the VMware database that you want to collect data from. 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 Feed Subfolder on page 2-14). This is the CSR file that is processed by the Scan program. This parameter is included as an identifier in the CSR file.</td>
</tr>
<tr>
<td>OutputFolder</td>
<td>The process definition folder for the collector. The output folder is defined by the Process id attribute in the job file. For example, if the Process id=“VMware”, the output folder is VMware.</td>
</tr>
<tr>
<td>MSAccessDBLocation</td>
<td>(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.</td>
</tr>
<tr>
<td>DataSourceID</td>
<td>(optional) The ITUAM 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 ITUAM Data Source that is set as the Web/Collector default in ITUAM Administrator is used. The default ITUAM Data Source should point to the ITUAM database.</td>
</tr>
</tbody>
</table>

Table 3-5 • VMware.wsf Parameters (Continued)

Running the VMware Collector

Use ITUAM Job Runner to run the VMware collector as described in Running ITUAM Job Runner on page 2-120.
Storage Data Collectors

This chapter contains instructions for setting up and running the ITUAM Data Collector for disk storage. You should have a good understanding of the ITUAM Data Collectors system architecture as described in the ITUAM Data Collectors Architecture section beginning on page 2-4 before continuing with the collector-specific information in this chapter.

Windows Disk Data Collector ................................................................. 4-2
Identifiers and Resources Collected by the Windows Disk Collector .................. 4-2
Setting Up the Windows Disk Collector ................................................. 4-3
Running the Windows Disk Collector .................................................. 4-9
Windows Disk Data Collector

The 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 CSR files. The files are produced by the collector's executable program, CIMSWinDisk.exe. If you installed ITUAM in the default location, this program is in C:\Program Files\IBM\ITUAM\Collectors\CIMSWinDisk.

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

Identifiers and Resources Collected by the Windows Disk Collector

By default, the 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.

<table>
<thead>
<tr>
<th>Identifier Name or Resource Description in ITUAM</th>
<th>Assigned Rate Code in ITUAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifiers</strong></td>
<td></td>
</tr>
<tr>
<td>Feed (defined by the Feed parameter in the Windows Disk collector job file [see page 4-8])</td>
<td>—</td>
</tr>
<tr>
<td>Folder (defined by the PathToScan parameter in the Windows Disk collector job file [see page 4-8])</td>
<td>—</td>
</tr>
<tr>
<td><strong>Resources</strong></td>
<td></td>
</tr>
<tr>
<td>MS Windows Disk Folder Usage in GB</td>
<td>DISKSIZE (GB days)</td>
</tr>
<tr>
<td>MS Windows Files in Folder</td>
<td>DISKFILE</td>
</tr>
</tbody>
</table>

Table 4-1 • Default Windows Disk Identifiers and Resources
Setting Up the Windows Disk Collector

On the central ITUAM Data Collectors server, set up an XML job file for the Windows Disk collector as described in Creating Job Files on page 2-27. The following is a sample job file. This example scans drive C of Server1 and Server2. The job file is running on Server1.

```xml
<?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="true" smtpSendJobLog="true" smtpServer="mail.ITUAMCustomerCompany.com" smtpFrom="ITUAM@ITUAMCustomerCompany.com" smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com" stopOnProcessFailure="false">
    <Process id="CIMSWinDisk" description="Multistep Processing for Windows Disk Collector" active="true">
      <Defaults>
        <Default programName="CIMSACCT" accCodeConvTable="C:\CIMS\AccountCodeTable\AccountCodeTable\AcctTable-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">
          <CIMSWinDisk filename="%ProcessFolder%\CIMSWinDisk.xml" overwrite="true">
            <CIMSServers version = "1.0">
              <Collectors>
                <Collector name="CIMSWinDisk" instanceName="Server1-C" instanceDescription="Scan of Server1 C" 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="C:"/>
                    <Parameter name="Units" value="GB"/>
                    <Parameter name="NumberOfLevels" value="1"/>
                  </Parameters>
                </Collector>
              </Collectors>
            </CIMSServers>
          </CIMSWinDisk>
        </Step>
      </Steps>
    </Process>
  </Job>
</Jobs>
```
**Storage Data Collectors**

*Windows Disk Data Collector*

```xml
</Collectors>
</CIMSCollectors>
</CIMSWinDisk>
<Parameters>
  <Parameter UseStandardParameters="false"/>
  <Parameter XMLFileName="%ProcessFolder%\CIMSWinDisk.xml"/>
  <Parameter CollectorName="CIMSWinDisk"/>
</Parameters>

<Step id="Server2 Collection" description="Server1 CIMSWinDisk" type="ConvertToCSR" programName="CIMSWinDisk\CIMSWinDisk.exe" programType="console" active="true">
  <CIMSWinDisk filename="%ProcessFolder%\CIMSWinDisk.xml" overwrite="true">
    <CIMSCollectors version = "1.0">
      <Collectors>
        <Collector name="CIMSWinDisk" instanceName="Server2-C" instanceDescription="Scan of Server2 C" 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>
  </CIMSWinDisk>
<Parameters>
  <Parameter UseStandardParameters="false"/>
  <Parameter XMLFileName="%ProcessFolder%\CIMSWinDisk.xml"/>
  <Parameter CollectorName="CIMSWinDisk"/>
</Parameters>

<Step id="Scan" description="Scan CIMSWinDisk" type="Process" programName="Scan" programType="net" active="true">
  <Parameters>
    <Parameter retainFileDate="false"/>
    <Parameter allowMissingFiles="false"/>
    <Parameter allowEmptyFiles="false"/>
  </Parameters>
</Step>
```
Note that the **ConvertToCSR steps contain the child elements** `CIMSWinDisk` and `Parameters`.

When CIMS Job Runner is run, the `CIMSWinDisk` element dynamically creates an XML file that contains parameters required by the Windows Disk collector. For a description of the elements and attributes for this file, see *Defining the Windows Disk Collector Attributes* on page 4-6.

The `Parameters` element provides parameters for the `ConvertToCSR step`. For a description of these parameters, see page 2-86.

For a description of all other elements and attributes in the process, see *Job File Structure* on page 2-44.
### Defining the Windows Disk Collector Attributes

Table 4-2 describes the attributes for the CIMSWinDisk element and its child elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMSWinDisk</td>
<td>fileName</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If you provide a full path, the path must be an existing path unless you include the attribute <code>createPath=&quot;true&quot;</code>.</td>
</tr>
<tr>
<td></td>
<td>overwrite</td>
<td>Specifies whether the file should overwrite an existing file. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (the existing file is overwritten)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (the file is not overwritten and the step fails)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
<tr>
<td></td>
<td>autoRemove</td>
<td>Specifies whether the file should be automatically removed after the step has executed. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (the file is removed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (the file is not removed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td>createPath</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (the path is created)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (the path is not created)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;false&quot;.</td>
</tr>
</tbody>
</table>

<p>| Table 4-2 • CIMSWinDisk Attributes |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector</td>
<td>Collector name</td>
<td>The collector name. <em>Do not change this parameter.</em></td>
</tr>
<tr>
<td></td>
<td>instanceName</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>instanceDescription</td>
<td>A description of the instance for the collector.</td>
</tr>
<tr>
<td></td>
<td>active</td>
<td>Specifies whether the instance is included in processing. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;true&quot; (the instance is processed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;false&quot; (the instance is not processed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is &quot;true&quot;.</td>
</tr>
<tr>
<td>Parameter</td>
<td>LogDate</td>
<td>The 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. To include the actual date that the data was collected, you need to use %RNDATE% as the LogDate parameter and you need to include the parameter LogDate=&quot;RNDATE&quot; at the job or process level in the job file (see the example on page 4-3).</td>
</tr>
</tbody>
</table>

**Retention**

This attribute is for future use.

Table 4-2 • CIMSWinDisk Attributes (Continued)
## Storage Data Collectors

### Windows Disk Data Collector

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Description/Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter (continued)</td>
<td>Feed</td>
<td>The name of the drive or folder that you want to collect disk space usage for. 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 Feed Subfolder on page 2-14). This is the CSR file that is processed by the Scan program. This parameter is included as an identifier in the CSR file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OutputFolder</td>
</tr>
</tbody>
</table>
|               |            | PathToScan         | Valid values for this attribute are:  
|               |            | • The drive or folder one level above the folder information you want to collect. For example, "PathToScan" value="\Server1\C$" collects data for all top level folders under the C share.  
|               |            | Note that \\Server1\C$ is an example UNC path, which is recommended.  
|               |            | • 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.  
|               |            | Note: To scan a shared drive, the Windows user ID used to log on to the computer running the Windows Disk collector must have authority to scan the share. |

Table 4-2 • CIMSWinDisk Attributes (Continued)
Using Smart Scan With the 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 Windows Disk collector is provided in an external XML 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 to 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 2-40.

Running the Windows Disk Collector

Use CIIMS Job Runner to run the Windows Disk collector as described in Running ITUAM Job Runner on page 2-120.
Storage Data Collectors

Windows Disk Data Collector
Tivoli Decision Support for z/OS (TDSz) Collector

This chapter contains instructions for setting up and running ITUAM Data Collector for TDSz. You should have a good understanding of the ITUAM Data Collector system architecture as described in the ITUAM Data Collectors Architecture section beginning on page 2-4 before continuing with the collector-specific information in this chapter.

About TDSz Data Collection ........................................................................... 5-2
Installing and Configuring DB2 Connect .......................................................... 5-2
Setting Up Conversion Definitions ................................................................. 5-4
About ITUAM Conversion Engine and Conversion Builder ............................... 5-5
Working With Conversion Definitions ............................................................ 5-5
Running ITUAM Conversion Engine from ITUAM Conversion Builder .......... 5-14
Creating a Job File for the TDSz Collector ...................................................... 5-15
Collection Step Parameters ............................................................................ 5-17
Running the TDSz Collector Job File .............................................................. 5-18
About TDSz Data Collection

TDSz is designed to be a highly effective, adaptable performance reporting tool for z/OS that can help you to successfully track system performance and resource use, anticipate and solve problems, and plan for future resource needs. TDSz uses a DB2® UDB for z/OS V8 database to provide a central repository for easy access to enterprise-wide IT information. This information can be valuable in performance reporting, service level management, and accounting.

The ITUAM Data Collector for TDSz enables you to collect z/OS data for processing and reporting in the Windows environment. The TDSz collector gathers data contained in the DB2 database and produces CSR files that can be processed by ITUAM. You can then view the processed data in report, spreadsheet, and graph format via intranet or Internet. These Web-base report provide multi-level drill down capabilities to view detailed cost and usage information in reports.

To collect TDSz data, you need to perform the following steps as described in the following sections:

- Install and Configure DB2 Connect
- Set up the conversion definition files provided with ITUAM
- Create a job file to collect and process the data
- Run ITUAM Job Runner

Installing and Configuring DB2 Connect

For a detailed description of how to install and configure DB2 Connect, refer to the IBM DB2 Connect for Windows Installation Guide.

The following is an example of how to configure DB2 Connect. The requirements for your site might be different than this example.

1. In Windows, click Start ➤ Programs ➤ IBM DB2 ➤ Set-up Tools ➤ Configuration Assistant.

2. In Configuration Assistant, click Selected ➤ Add Database Using Wizard.

   The Add Database Wizard appears.

3. On the Select how you want to set up a connection page, click Manually configure a connection to a database, and then click Next.

4. On the Select a communications protocol page, click TCP/IP, and then click Next.
On the **Specify TCP/IP communication parameters** page, complete the following, and then click **Next**:

- **Host name.** Type the host name, IP address, or IP name where the target database resides.

- **Service name.** Type the service name of the server database manager instance or leave this box blank.

- **Port number.** Type the port number associated with the DB2 server instance that contains the target database.

To determine the correct host name and port number, refer to the DDF configuration which is displayed in the z/OS startup messages as shown in the following example:

```
06.39.08 STC09640 DSNL0041 @DDF START COMPLETE 872
872 LOCATION KSCDB201
872 LU USIBMKS.KSCDB201
872 GENERICLU -NONE
872 DOMAIN demomvs.db2.ibm.com
872 TCP PORT 5663
872 RES PORT 5463
```

In this example, the domain is demomvs.db2.ibm.com and the port is 5663.

**On the Specify the name of the database to which you want to connect** page, type the database name and alias, and then click **Next**. Note that the database name is not a database within the DB2 UDB for z/OS instance. It is the location name specified in the DDF configuration. In the preceding startup message, the database name is KSCDB201.

**On the Register this database as a data source** page, accept the default values, and then click **Next**.

**On the Specify the node options** page, click **OS/390 or z/OS** in the Operating System box. In the Instance name box, type the name of the instance where the database is located. Click **Next**.

**On the Specify the system options** page, type the system name in the System name box. This is the computer, server system, or workstation where the target database is located. Click **Next**.

**On the Specify the security options** page, click **Use authentication value in server’s DBM Configuration**. Click **Finish**.

The Add Database Confirmation message box appears. You can use this box to close the wizard, add another database, change the database configuration that you just added, or test the connection to the database (recommended).

To view the database, click **Start ➤ Programs ➤ IBM DB2 ➤ General Administration Tools ➤ Control Center**.
Setting Up Conversion Definitions

The **IBMTDS** folder in ...Program Files\IBM\ITUAM\Collectors, contains sample pre-defined conversion definitions. A conversion definition is a file that defines the format of the data source (in this case a DB2 database) as well as the data that will appear in the output CSR file.

Table 5-1 shows each of the pre-defined conversion definitions in the **IBMTDS** folder and the corresponding RAF table that the definition supports. These tables were chosen because they contain chargeback data. If you want to collect data from other TDSz tables, you can create a conversion definition for any table.

<table>
<thead>
<tr>
<th>Conversion Definition</th>
<th>TDSz Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>RafAddrLogDef.txt</td>
<td>RAFADDRLOG</td>
</tr>
<tr>
<td>RafBatchDef.txt</td>
<td>RAFBATCH</td>
</tr>
<tr>
<td>RafCICSDef.txt</td>
<td>RAFCICS</td>
</tr>
<tr>
<td>RafDASDDef.txt</td>
<td>RAFDASD</td>
</tr>
<tr>
<td>RafDB2Def.txt</td>
<td>RAFDB2</td>
</tr>
<tr>
<td>RafHSMBKupDef.txt</td>
<td>RAFFHSMBKUP</td>
</tr>
<tr>
<td>RafHSMMigrDef.txt</td>
<td>RAFFHSMMIGR</td>
</tr>
<tr>
<td>RafIMSDef.txt</td>
<td>RAFIMS</td>
</tr>
<tr>
<td>RafJobLogDef.txt</td>
<td>RAFJOBLOG</td>
</tr>
<tr>
<td>RafNetSes4xDef.txt</td>
<td>RAFFNETSES4X</td>
</tr>
<tr>
<td>RafNetSes20Def.txt</td>
<td>RAFFNETSES20</td>
</tr>
<tr>
<td>RafNetvDef.txt</td>
<td>RAFFNETIV</td>
</tr>
<tr>
<td>RafSesLogDef.txt</td>
<td>RAFSESLOG</td>
</tr>
<tr>
<td>RafSTCDef.txt</td>
<td>RAFSTC</td>
</tr>
<tr>
<td>RafTSDDef.txt</td>
<td>RAFTSO</td>
</tr>
</tbody>
</table>

*Table 5-1  • Default Conversion Definitions for TDSz Data*

These sample conversion definitions do not extract all of the data from the corresponding tables. That is, not all data in the tables is defined as identifiers and resources to be included in the CSR file. If you want to define other identifiers and/or resources, you need to modify the conversion definition.

**Note**  • If you add resources to a conversion definition, you need to add the rate codes for the resources to the CIMSRate table. For instructions on creating and maintaining rate codes, refer to the **IBM Tivoli Usage and Accounting Manager Administrator’s Guide**.
About ITUAM Conversion Engine and Conversion Builder

ITUAM Conversion Engine is a utility that is used to convert data into CSR files. The TDSz collector requires that the ITUAM Conversion Engine was installed with the ITUAM installation. (The default is to install the engine.)

ITUAM Conversion Builder is a GUI application that you can use to create the conversion definitions that are fed into CIMS Conversion Engine.

Working With Conversion Definitions

You can modify the sample conversion definitions using the ITUAM Conversion Builder application (see Viewing and Modifying the Conversion Definition in ITUAM Conversion Builder) or you can modify definition files using a text editor such as Notepad (see Viewing and Modifying the Conversion Definition in Notepad on page 5-13).

Viewing and Modifying the Conversion Definition in ITUAM Conversion Builder

To modify a conversion definition in ITUAM Conversion Builder:

1. Click Start ➔ Programs ➔ ITUAM ➔ ITUAM Conversion Builder.
2. Click File ➔ Open Conversion Definition.
3. Browse to the ...IBM/ITUAM/Collectors/IBMTDS folder and open the process definition that you want to modify.

The data in the conversion definition populates the tabs in ITUAM Conversion Builder. The following sections provide a brief description of the key fields and options on each tab. For a detailed description of all fields and options, see Appendix B, Conversion Definition File Description.

The examples in the following section reflect the conversion definition RAFDB2Def.txt as it appears by default. You can edit this and all conversion definitions as needed.
Input Tab

The Input tab defines the input source used by the conversion definition.

![Input Tab](image)

Note the following fields:

- **Input Type.** The setting ODBC Query specifies that the data is being collected from a database using the database query specified in the ODBC SQL Query field. Do not change the input type.

- **ODBC Data Source.** This is the name of the ODBC Data Source for the DB2 database that you are collecting from.

- **ODBC SQL Query.** This is the SQL query that is run against the TDSz table. Note the following:
  
  - The field CPUSEC is defined as FLOAT in the DB2 for z/OS database. You must use the decimal function to turn this field into a character representation or the data will return in scientific notation, which is not supported by ITUAM Processing Engine.
  
  - Only those table fields specified in the query are processed. To process all fields in the table:
    
    a. Change the query to: SELECT * from tds.rafdb2 (where the table depends on the conversion definition).

    b. Click the Fields tab, and then click Populate Field Listing Using Query.

For a description of all of the fields on this tab, see Input Parameters on page B-3.
Output Tab

The Output tab defines the output file produced by the conversion definition.

Figure 5-2 • Output Tab

Note the following fields:

- **Output Filename.** This is the path and name of the output CSR file. When you run the process definition from a job file, the path is defined by the job file and this value is ignored.

- **Output Record Type.** This field defines the type of output file that is created. Always select CSR Record. The other record type options are deprecated.

For a description of all of the fields on this tab, see *Output Parameters* on page B-7.
Fields Tab

The Fields tab defines the TDSz table fields that are used to produce the data in the output CSR file.

The fields shown on this tab should be the same as those in the SQL query shown on the Input tab. If you change the query, click the Populate Field List Using Query button.

The value in the Type (Date/Time) field defines how the date will appear in the CSR file.

For a description of all of the fields on this tab, see Field Definition Parameters on page B-9.

![Figure 5-3 • Fields Tab](image-url)
Identifiers Tab

The Identifiers tab defines the TDSz table fields to be used as identifiers in the output CSR file.

For a description of all of the fields on this tab, see Identifier Parameters on page B-15.

Figure 5-4 • Identifiers Tab
Resources Tab

The Resources tab defines the TDSz table fields to be used as resources in the output CSR file. The correlating rate code for each field is defined in the Rate Code field.

You can use the default rate codes or change the code. The default rate codes are pre-loaded in the CIMSRate table. If you want to use these rate codes, you can simply modify the values for the rate codes as needed.

If you change the rate codes, you need to add the codes to the CIMSRate table.

For instructions on creating and maintaining rate codes, refer to the IBM Tivoli Usage and Accounting Manager Administrator's Guide.

For a description of all of the fields on this tab, see Resource Parameters on page B-16.

Figure 5-5 • Resources Tab
**Date/Time Tab**

The **Date/Time** tab defines the start and end date and time that appear in the output file. In this example, the dates are derived from the TDSz table field **DATE**.

For a description of all of the fields on this tab, see **Date/Time Parameters** on page B-19.

![Figure 5-6 • Date/Time Tab](image-url)
Shifts Tab

The Shifts tab defines whether shift processing is enabled. Rate shifts allow you to set different rates based on the time of day. Shift processing is optional and is not enabled in the sample conversion definition files.

For a description of all of the fields on this tab, see Shift Parameters on page B-21.

![Shifts Tab](image)

**Figure 5-7 • Shifts Tab**
Viewing and Modifying the Conversion Definition in Notepad

A conversion definition file 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.

You can edit a conversion definition directly from Notepad or another text editor. The following shows the conversion definition RafDB2Def.txt open in Notepad.

```plaintext
[Control]
Description=TDS Table
InputFile=NDDB201
OutputFile=C:\program files\ibm\ituam\sample processes\ibmtds\rafdb2.TXT
OutRecType=CBS
ProcessType=ODBCQUERY
Delimiter=COMMA
RecDelimiter=NEWLINE
InitialSkipLineCnt=0
TextQualifier=DOUBLEQUOTE
RecDateLo=DATE
RecTimeLo=ENTIRE
RecTimeHi=ENTIRE
UnivHdr=TDSzDB2
ShiftsEnabled=NO
OdbcQuery=SELECT decimal(elapstim,20,6) as etime, decimal(cpusec,20,6) as cpu, \n decimal(crdbase) as crdbase, decimal(crint) as crindex, \n decimal(crtabl) as crtable, decimal(crtabsp) as crtabsp, \n decimal(deletecalls) as dcalls, decimal(fetchcalls) as fcalls, \n decimal(insertcalls) as icalls, \n date, sysid, subsysid, account, conntype, planname
FROM tds.rafdb2;
RateOdbcDsn=
WriteStandardServerIdentifiers=YES
RecDateHi=DATE
[Layout]
Field1=ETIME
Field2=CPU
Field3=CRDBASE
Field4=CRINDEX
Field5=CRTABLE
Field6=CRTABSP
Field7=DCALLS
Field8=FCALLS
IDField1=SYSID
IDField2=SUBSYSID
IDField3=ACCOUNT
IDField4=CONNTYPE
IDField5=PLANNAME
RSField1=CPU RATECODE(ZZ32)
RSField2=ETIME RATECODE(ZZ34)
Field9=ICALLS
Field10=DATE TYPE(D-YYYY/MM/DD)
IDField11=SYSID
IDField12=SUBSYSID
IDField13=ACCOUNT
IDField14=CONNTYPE
IDField15=PLANNAME
RSField13=CRTABSP RATECODE(TDB2CRTS)
RSField14=CRINDEX RATECODE(TDB2CRIN)
RSField15=CRTABLE RATECODE(TDB2CRTB)
RSField16=CALLS RATECODE(TDB2CALS)
RSField17=DCALLS RATECODE(TDB2DCAL)
RSField18=FCALLS RATECODE(TDB2FCAL)
RSField19=ICALLS RATECODE(TDB2ICAL)
```

IBM Tivoli Usage and Accounting Manager Data Collectors for Microsoft Windows User’s Guide 5-13
Running ITUAM Conversion Engine from ITUAM Conversion Builder

Once you have created a conversion definition for a TDSz table, you can run ITUAM Conversion Engine directly from ITUAM Conversion Builder to ensure that the CSR file contains the data that you want.

To run Conversion Engine, click File ➤ Run Conversion. The CSR file is created and placed in the location specified on the Output tab (see Output Tab on page 5-7).

**Note** • ITUAM Conversion Builder is not intended to be used for daily data conversion. Daily conversion should be performed in batch as described in Running ITUAM Data Collectors on page 2-119.
Creating a Job File for the TDSz Collector

On the central ITUAM Data Collectors server, set up an XML job file for the TDSz collector as described in Creating Job Files on page 2-27. The following is a sample job file. In this example, data is collected from the RAFADDRLOG, RAFBATCH, and RAFDB2 tables.

```xml
<?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="true"
    smtpSendJobLog="true"
    smtpServer="mail.ITUAMCustomerCompany.com"
    smtpFrom="ITUAM@ITUAMCustomerCompany.com"
    smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
    stopOnProcessFailure="false">
    <Process id="IBMTDS"
      description="Process for Windows Process Collection"
      active="true">
      <Defaults>
        <Default programName="CIMSACCT"
          accCodeConvTable="C:\CIMS\AccountCodeTable\AccountCodeTable\AcctTab1-zOS.txt"/>
      </Defaults>
      <Steps>
        <Step id="RAFADDRLOG Collection"
          description="RAFADDRLOG Collection"
          type="ConvertToCSR"
          programName="IBMTDS\TDSz.wsf"
          programType="wsf"
          active="true">
          <Parameters>
            <Parameter Feed="ADDRLOG"/>
            <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafAddrLogDef.txt"/>
          </Parameters>
        </Step>
        <Step id="RAFBATCH Collection"
          description="RAFBATCH Collection"
          type="ConvertToCSR"
          programName="IBMTDS\TDSz.wsf"
          programType="wsf"
          active="true">
          <Parameters>
            <Parameter Feed="BATCH"/>
            <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafBatchDef.txt"/>
          </Parameters>
        </Step>
      </Steps>
  </Job>
</Jobs>
```
Creating a Job File for the TDSz Collector

```xml
<Step id="RAFDB2 Collection"
     description="RAFDB2 Collection"
     type="ConvertToCSR"
     programName="IBMTDS\TDSz.wsf"
     programType="wsf"
     active="true">
  <Parameters>
    <Parameter Feed="DB2"/>
    <Parameter ConvEngDefName="C:\Program Files\IBM\ITUAM\Collectors\IBMTDS\RafDB2Def.txt"/>
  </Parameters>
</Step>

<Step id="Scan"
     description="Scan"
     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"
      type="Process"
      programName="SingleProcessStep"
      programType="com"
      active="true">
</Step>

<Step id="DatabaseLoad"
      description="Database Load"
      type="Process"
      programName="DBLoad"
      programType="com"
      active="true">
</Step>

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

The following parameters appear in each collection step (type="ConvertToCSR") in the SampleNightly_TDS.xml job file. These parameters are passed to the TDSz.wsf script in the Collectors\IBMTDS folder.

- **Feed**
  The name of the server that contains the TDSz database that you want to collect data from.
  A subfolder with the same name as the server is automatically created in the process definition folder. This subfolder is used to store the initial CSR file that is created by the collector (see Feed Subfolder on page 2-14). This is the CSR file that is processed by the Scan program.
  This parameter is included as an identifier in the CSR file.

- **ConvEngDefName**
  The location of the conversion definition file to be used.

The following parameters do not appear in the ConvertToCSR steps, but are also required by the TDSz.wsf script. When these parameters are not provided in the job file, the default values are passed.

- **LogDate**
  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.) For example, if you run the job file on May 1, 2006 with a log date of PREDAY, the CSR file is named 20060430.txt.
  The start and end dates that appear in the CSR file records are defined by the conversion definition file that is specified by the ConvEngDefName parameter. For more information, see Date/Time Tab on page 5-11.
  For more information about using a log date, including valid log date values, see Specifying Log Dates for Collection on page 2-4.

- **RetentionFlag**
  This parameter is for future use.

- **OutputFolder**
  The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.
  By default, the output folder is defined by the Process id attribute in the job file. For example, if the Process id="IBMTDS", the output folder is IBMTDS.
Running the TDSz Collector Job File

Use ITUAM Job Runner to run the TDSz collector as described in Running ITUAM Job Runner on page 2-120.
This chapter contains instructions for setting up and running collection of the CSR files produced by ITUAM Data Collectors for UNIX and Linux. You should have a good understanding of the data collection system architecture described in the ITUAM Data Collectors Architecture section beginning on page 2-4 before continuing with the information in this chapter.

About UNIX/Linux Data Collection ................................................................. 6-2
Setting Up Collection of UNIX/Linux CSR Files ................................................. 6-2
Running the UNIX/Linux CSR File Collection Process ....................................... 6-4
About UNIX/Linux Data Collection

ITUAM Data Collectors for UNIX and Linux produce CSR files that can be processed by ITUAM. This chapter discusses how to collect and process the CSR files for input into the ITUAM database.

The CSR files are sent via FTP or Secure Shell from the system running ITUAM Data Collectors for UNIX and Linux to the appropriate process definition folder on the ITUAM system. ITUAM includes default process definition folders named UnixOS and UnixFS that may be used to store these files. For more information about process definition folders, see page 2-13.

CSR files must be sent to a feed subfolder within the process definition folder (see Feed Subfolder on page 2-14). The feed subfolder might represent the server from which the CSR files are sent. For example, if the server ABCServer is sending CSR files containing UNIX or Linux file system data, you might create the feed subfolder ABCServer in the UnixFS process definition folder. The CSR files are named yyyyymmdd.txt.

For more information creating and transferring UNIX/Linux CSR files, refer to the IBM Tivoli Usage and Accounting Manager Data Collectors for UNIX and Linux User’s Guide.

The following section describes how to set up the collection process for the CSR files.

Setting Up Collection of UNIX/Linux CSR Files

On the central ITUAM Data Collectors server, set up an XML job file as described in Creating Job Files on page 2-27. 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.

```xml
<?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="true" smtpSendJobLog="true"
smtpServer="mail.ITUAMCustomerCompany.com" smtpFrom="ITUAM@ITUAMCustomerCompany.com"
smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com" stopOnProcessFailure="false">
    <Process id="UnixOS" description="Process for Unix OS Collection" active="true">
      <Steps>
        <Step id="Scan UnixOS" description="Scan UnixOS">
         ...
        </Step>
      </Steps>
    </Process>
  </Job>
</Jobs>
```
Processing Data from ITUAM Data Collectors for UNIX/Linux

Setting Up Collection of UNIX/Linux CSR Files

```xml
<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/Linux CSR File Collection Process

To run the collection process for CSR files produced by ITUAM Data Collectors for UNIX and Linux, see Running ITUAM Job Runner on page 2-120.
ITUAM Aggregation Engine API

About ITUAM 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
About ITUAM Aggregation Engine

ITUAM Aggregation Engine (CIMSAggregation.dll) is a COM object that aggregates the records in the log file by identifier values. ITUAM 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 CSR file.

Aggregation reduces the amount of data from a log file that must be processed by ITUAM Processing Engine, thus reducing processing time. This is especially beneficial for log files that are created daily and contain gigabytes of data.

ITUAM 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. ITUAM 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,20061031,20061031,11:02:43,,2,User,BERT,Machine,ACME1,1,SODA,1
ACMESODA,20061031,20061031,11:02:57,,2,User,JANICE,Machine,ACME1,1,SODA,1
ACMESODA,20061031,20061031,12:05:34,,2,User,JANICE,Machine,ACME1,1,SODA,1
ACMESODA,20061031,20061031,12:10:05,,2,User,RANDY,Machine,ACME1,1,SODA,1

Once an aggregation key is created, resource values passed to ITUAM 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. ITUAM 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 ITUAM Aggregation Engine, the engine writes a CSR file.

About the Aggregation Algorithm

The base aggregation algorithm used by ITUAM Aggregation Engine is the Repeated Scanning algorithm\(^1\). 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.

ITUAM Aggregation Engine API

**Processing the Log File**

ITUAM 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 ITUAM 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 ITUAM 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, the these methods are used. For these parameters, a time can be specified as part of the date.

If the 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 ITUAM Aggregation engine does not process modified records. To write modified records to a CSR file, use a script. The class CSRWriter in the CIMSLib.wsf script is provided 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 CSR 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#00003(Identifier Value [String])
AggList (0, 6) = UserId(Identifier Name [String])
AggList (0, 7) = User#00003(Identifier Value [String])
AggList (0, 8) = FileNumber(Identifier Name [String])
AggList (0, 9) = FileNum(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.
**ITUAM Aggregation Engine API**

### DateAggregation
Returns or sets the date field to aggregate on.

**Syntax**

```
object.DateAggregation [=value]
```

**Parameter**

**Value**

A value specifying the date fields to aggregate on. Valid values are `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 CSR record will contain a minimum of the start date value and the maximum of the end date value.

If `EDateAggregation_StartDate` is specified, the CSR record start date/time fields will contain the minimum of the start date value. The CSR record end date/time fields will contain the end date/time from the first record.

If `EDateAggregation_EndDate` is specified, the CSR 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 CSR 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 CSR 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 ITUAM 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**

ITUAM 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, ITUAM 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 ITUAM Aggregation Engine will use to store aggregates.

**Syntax**

```plaintext
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**

```plaintext
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**

```plaintext
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**

```plaintext
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 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 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 an ITUAM result code indicating whether the entry specified was added successfully. The ITUAM 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**

```vbnet
object.AddEntries(ByVal NumberOfEntries As Variant, _
                   ByVal IdentifierValueList() As Variant, _
                   ByVal ResourceValueList() As Variant, _
                   Optional ByVal DateStartList As Variant, _
                   Optional ByVal 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 ITUAM result code indicating whether all of the entries specified were added successfully. The ITUAM 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**

```csharp
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**

```csharp
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 may 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 ITUAM 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 `CIMExceptionFile.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**

```vbnet
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**

```vbnet
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.
Conversion Definition File Description

This appendix describes the parameters in conversion definition file. The information in this appendix is general to all collectors. For information on how to set up a conversion definition file for a specific collector, refer to the documentation for that collector in a preceding chapter.

Overview ........................................................................................................... B-2
Conversion Definition Components ................................................................. B-3
Input Parameters ............................................................................................... B-3
Output Parameters ............................................................................................. B-7
Field Definition Parameters .............................................................................. B-9
Identifier Parameters ....................................................................................... B-15
Resource Parameters ....................................................................................... B-16
Date/Time Parameters ...................................................................................... B-19
Shift Parameters ............................................................................................... B-21
Overview

Some ITUAM Data Collectors require a conversion definition to convert the data in an input data source. A conversion definition is a file that defines the format of the data source as well as the data that will appear in the output CSR file.

Conversion definitions are fed into ITUAM Conversion Engine to convert the data and product CSR files.

You can view and modify conversion definitions in either of the following ways:

- Using the ITUAM Conversion Builder GUI.
- Using a text editor such as Notepad.

The following sections provide descriptions for each of the parameters in a conversion definition as they appear in the ITUAM Conversion GUI and as they appear in a text editor.
Conversion Definition Components

Input Parameters

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description= &lt;description of this definition file&gt;</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Type</th>
<th>ProcessType=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delimited ASCII Text File (default)</td>
<td>DELIMITED</td>
<td>The type of data to be processed: data in an ASCII text file or data extracted from a database query. 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. The input type determines which of the following processing options appear. If the input type is an ASCII text file, continue to ASCII Text File on page B-4. If the input type is a database query, skip to Database Query on page B-6.</td>
</tr>
<tr>
<td>Fixed-length ASCII Text File</td>
<td>FIXED</td>
<td></td>
</tr>
<tr>
<td>ODBC Query</td>
<td>ODBCQUERY</td>
<td></td>
</tr>
<tr>
<td>Microsoft Access Database Query</td>
<td>MSACCESS</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-1  • Input Parameters
<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASCII Text File</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Filename</td>
<td>InputFile=&lt;path and file name of input file&gt;</td>
<td>This is the path and name of the input usage metering file. The maximum path that can be specified is approximately 250 bytes.</td>
</tr>
<tr>
<td>Record Delimiter</td>
<td>RecDelimiter =</td>
<td>The character used to delimit records (normally NEWLINE).</td>
</tr>
<tr>
<td>■ NEWLINE (default)</td>
<td>■ NEWLINE</td>
<td>If fields are terminated by a new line, then set the record delimiter to BLANKLINE.</td>
</tr>
<tr>
<td>■ BLANKLINE</td>
<td>■ BLANKLINE</td>
<td>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.</td>
</tr>
<tr>
<td>■ FORMFEED</td>
<td>■ FORMFEED</td>
<td></td>
</tr>
<tr>
<td>Field Delimiter (does not appear for Fixed-length input type)</td>
<td>Delimiter=</td>
<td>The character used to delimit fields in a usage metering file.</td>
</tr>
<tr>
<td>■ COMMA (default)</td>
<td>■ COMMA</td>
<td>A field delimiter is required only for delimited files.</td>
</tr>
<tr>
<td>■ TAB</td>
<td>■ TAB</td>
<td>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 (/).</td>
</tr>
<tr>
<td>■ SEMICOLON</td>
<td>■ SEMICOLON</td>
<td></td>
</tr>
<tr>
<td>■ COLON</td>
<td>■ COLON</td>
<td></td>
</tr>
<tr>
<td>■ NEWLINE</td>
<td>■ NEWLINE</td>
<td></td>
</tr>
<tr>
<td>■ SPACE</td>
<td>■ &lt;any character literal&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-1 • Input Parameters (Continued)
### Conversion Definition File Description

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Field Qualifier (does not appear for Fixed-length input type)</td>
<td>TextQualifier=</td>
<td>The character used for fields with embedded delimiter characters. For example, if the field delimiter is COMMA and the field value is &quot;1,345&quot; then the text field qualifier is DOUBLEQUOTE. The quotation marks in this case mark the beginning and ending of the field value. Quote indicates a single quote qualifier. A text field qualifier is required only for delimited files. You can select a qualifier from the list, or you can enter the ASCII keyboard character(s) for other delimiters. For example, an asterisk (*).</td>
</tr>
<tr>
<td>DOUBLEQUOTE (default)</td>
<td>DOUBLEQUOTE</td>
<td></td>
</tr>
<tr>
<td>QUOTE</td>
<td>QUOTE</td>
<td></td>
</tr>
<tr>
<td>NONE</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>&lt;any character literal&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Skip Initial Lines | InitialSkipLineCnt = n | 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. This is useful in situations where there are a number of header lines preceding the actual data. The default is 0 (skip no lines). |

| Table 2-1 • Input Parameters (Continued) |
### Database Query

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
</table>
| ODBC Data Source (appears when ODBC Query input type is selected) | InputFile=<name of ODBC data source> | The ODBC Data Source for the database to be queried. The data source must be listed in the Windows ODBC Data Source Administrator. Do one of the following:  
- Type the data source name (the data source must be listed in the Windows ODBC Data Source Administrator).  
- Click Browse to the 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 OK.  
- If the data source is not listed in the dialog box, click the ODBC Data Source Administrator button, click the System DSN tab, and click Add to add the data source. |
| 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 2-1 • Input Parameters (Continued)*
# Output Parameters

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Filename</td>
<td>OutputFile=&lt;path and file name of generated output file&gt;</td>
<td>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 Feed Subfolder on page 2-14). The maximum path that can be specified is approximately 250 bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Record Type</th>
<th>OutRecType=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>■ CSR Record (default)</td>
<td>■ CBS</td>
<td>The record type in the output file. Always select CSR Record. The other record types (CIMS Transaction Record and CIMS Resource Record) are deprecated.</td>
</tr>
<tr>
<td>■ CIMS Transaction Record</td>
<td>■ TRANS</td>
<td></td>
</tr>
<tr>
<td>■ CIMS Resource Record</td>
<td>■ UNIV</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource Header (appears when CSR Record is selected)</th>
<th>UnivHdr=&lt;field name&gt;</th>
<th>The resource header defines the source of data. A resource header is not available in all usage metering files and is not required. Depending on whether the usage metering file contains a header, you can do the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ If the records within the file do not 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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ If the records within the file do contain a header, you can select the header field from the drop down list (if the field is entered in the Fields tab), type the field name, or leave the field blank (if you do not want the header to appear in the output file).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Table 2-2  •  Output Parameters |
## Conversion Definition File Description

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output standard server identifiers (appears when CSR Record is selected)</td>
<td><code>WriteStandardServerIdentifiers=</code></td>
<td>When this check box is selected, the identifiers <code>SourceName</code> and <code>SourceLine</code> are added to the output file. <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). <code>SourceLine</code> shows the line of the usage metering file that produced the record. 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).</td>
</tr>
<tr>
<td>Audit Code Default (appears when CIMS Transaction Record is selected)</td>
<td><code>AuditCodeDefault= &lt;string literal&gt;</code></td>
<td>A string that is used to hold a default audit code value (see page B-17). A default audit code is optional. 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.</td>
</tr>
</tbody>
</table>

Table 2-2 • Output Parameters (Continued)
**Field Definition Parameters**

The required parameters for fields 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 Parameters* on page B-19).

- For all other data sources, 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.

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Number</td>
<td>Fieldn</td>
<td>An incrementing sequence number used to uniquely identify a field.</td>
</tr>
<tr>
<td>Field Name</td>
<td>&lt;field name&gt;</td>
<td>The field name. The name must be a single word or abbreviation (for example, ACCTCD for account code).</td>
</tr>
<tr>
<td>Starting Column</td>
<td>COL(n)</td>
<td>The starting position for the field. The starting position number is required for fixed-length files. It is optional for delimited files.</td>
</tr>
<tr>
<td>Length</td>
<td>LEN(n)</td>
<td>The length of the field. The length is required for fixed-length files. It is optional for delimited files.</td>
</tr>
<tr>
<td>Implied Decimals</td>
<td>DEC(n)</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Table 2-3  Field Parameters*
Time and date fields require a TYPE declaration specifying the format of the time and date as they appear in the usage metering file.

The type format used for time and date fields is dependent on whether the time and date are fixed length or variable length.

### Time Fields

- **Fixed length.** 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.

  Fixed length time fields in a usage metering file do not 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.

  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-”.

  Examples: T-HHMMSS, T-HH:MM

---

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (Date/Time)</td>
<td>TYPE(date or time)</td>
<td>Time and date fields require a TYPE declaration specifying the format of the time and date as they appear in the usage metering file.</td>
</tr>
</tbody>
</table>

Table 2-3  Field Parameters (Continued)
Conversion Definition File Description

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (Date/Time) (continued)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Time Fields (continued)**

- **Variable length.** A variable length time format is one in which there is not a fixed number of digits for the time. For example, 12:34, 1:15 (no preceding 0), etc.

  Variable length time fields in a usage metering file require a separator character and the character must be included in the type format.

  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-”.

  Examples: T-H:M, T-H:M:S

**Date Fields**

- **Fixed length.** A fixed length date format is one in which there are a fixed number of digits for the date. For example, 12252006, 01012007, etc.

  Fixed length date fields in a usage metering file do not require a separator character. However, if the field includes a separator character, for example, 12/25/2006, you need to include the character in the type format. If the field does not include a separator, for example, 12252006, the separator character is optional.

  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-".

  Examples: D-YYYYMMDD, D-MM/DD/YYYY

Table 2-3 • Field Parameters (Continued)
### Date Fields (continued)

- **Variable length.** A variable length date format is one in which there is *not* a fixed number of digits for the date. For example, 12/25/2006, 1/01/2007 (no preceding 0), etc.

Variable length date fields in a usage metering file require a separator character and the character must be included in the type format.

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-".

Examples: D-Y/M/D, D-M/D/Y

If a year contains only two digits, the century is determined by the following:

- Years 0–29 are assumed to occur in the 2000s (2000–2029)
- Years 30–99 are assumed to occur in the 1900s (1930–1999)

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (Date/Time) (continued)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2-3 • Field Parameters (Continued)**
### Filter

**Definition File Statement**: `FILTERPATTERN`<br>**Description**: A regular expression or a literal that the field must match, otherwise the record is rejected.

Regular expressions are used frequently in some utilities and programming languages such as `grep`, `sed`, `awk`, and `Perl`.

The regular expression `FILTERPATTERN` is a subset of full regular expressions available in other tools and can consist of the following metacharacters:

- `^`–Matches to beginning of field
- `$`–Matches to end of field
- `*`–Matches zero or more occurrences of the preceding literal
- `.`–Matches any character
- `!`–If this is the first character in an expression, it negates the outcome of the regular expression. That is, the expression is not matched.

**Example**

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.

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td><code>FILTERPATTERN</code>&lt;br&gt;<code>(reg expression)</code></td>
<td>A regular expression or a literal that the field must match, otherwise the record is rejected.</td>
</tr>
</tbody>
</table>

---

**Table 2-3 • Field Parameters (Continued)**
**Conversion Definition File Description**

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse—Character</td>
<td>PARSECHAR (character)</td>
<td>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. Parse—Character and Parse—Word Number work together to parse a “word” from a string. For example, if you want only cimsnt.asp in the following example: <a href="http://www.cimslab.com/cimsnt.asp">http://www.cimslab.com/cimsnt.asp</a> The parse character is / and the parse word is the fourth word in the string as follows: Word 1=http: Word 2=null Word 3=www.cimslab.com Word 4=cimsnt.asp In this case, you enter a 4 in the Parse—Word Number box.</td>
</tr>
<tr>
<td>Parse—Word Number</td>
<td>PARSEWORD(n)</td>
<td>The number of the word in the string that should be split by the parse character (see the preceding example for Parse—Character) and returned as the field value. The character delimits the end of a word. 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 cimsnt.asp (there is no fifth word).</td>
</tr>
</tbody>
</table>

**Table 2-3 • Field Parameters (Continued)**
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 ITUAM.

**Table 2-3 • Field Parameters (Continued)**

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Field</td>
<td></td>
<td>Inserts a field above the selected field.</td>
</tr>
<tr>
<td>Remove Field</td>
<td></td>
<td>Removes the selected field.</td>
</tr>
<tr>
<td>Populate Field List Using Query (appears when a database query input type is selected)</td>
<td></td>
<td>Automatically populates the field list with fields from the database. You can then change the field names if needed.</td>
</tr>
</tbody>
</table>

### Identifier Parameters

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 ITUAM.

**Table 2-4 • Identifier Parameters**

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Number</td>
<td>IDFIELDn</td>
<td>An incrementing sequence number used to uniquely identify an identifier.</td>
</tr>
<tr>
<td>Field Name</td>
<td>&lt;field name&gt;</td>
<td>The field used as an identifier.</td>
</tr>
<tr>
<td>Insert Field</td>
<td></td>
<td>Inserts a field above the selected field.</td>
</tr>
<tr>
<td>Remove Field</td>
<td></td>
<td>Removes the selected field.</td>
</tr>
</tbody>
</table>
## Resource Parameters

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.

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Number</td>
<td>RSFILEDn</td>
<td>An incrementing sequence number used to uniquely identify a resource.</td>
</tr>
<tr>
<td>Field Name</td>
<td>&lt;field name&gt;</td>
<td>The field containing a resource.</td>
</tr>
<tr>
<td>Rate Code</td>
<td>RATECODE(code)</td>
<td>The rate code represents the resource units being reported by the field.</td>
</tr>
</tbody>
</table>

To enter a rate code for the field, click the **Rate Code** box. Click the (...) button and do one of the following:

- If a field within the usage metering file contains the rate code, click the field name.
- If the rate code is contained in the CIMSRate table of the ITUAM database (see **ODBC Data Source Name** on page B-18), click the existing rate code.
- 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.

**Important:** If you select a rate code from the usage metering file or create a new rate code, **you must add the rate code to the CIMSRate table**. Rate codes that do not appear in the CIMSRate table are not included in ITUAM invoices and other reports.

---

**Table 2-5 • Resource Parameters**
### ITUAM Conversion Builder Option

| Audit Code (appears when CIMS Transaction Record is selected as the output record type) | AUDITCODE(code) | A literal value specifying the audit code used to track this resource value. An audit code is optional. 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. To enter a audit code for the field, click the Audit Code box. Click the (...) button and do one of the following:
- If a field within the usage metering file contains the audit code, click the field name.
- If you want to use the default audit code entered in the Output tab, click DEFAULT.
- If you want to enter an audit code, type the code in the lower box. |
| Insert Field | Inserts a field above the selected field. |
| Remove Field | Removes the selected field. |

Table 2-5 • Resource Parameters (Continued)
### Conversion Definition File Description

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
</table>
| ODBC Data Source Name           | RateOdbcDsn=<name of database> | 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:  
  - Type the data source name (the data source must be listed in the Windows ODBC Data Source Administrator).  
  - Click Browse to the 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 OK.  
  - If the data source is not listed in the dialog box, click the ODBC Data Source Administrator button, click the System DSN tab, and click Add to add the data source. |
| 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 2-5 • Resource Parameters (Continued)*
Date/Time Parameters

Use the Date/Time tab to define the start and end date and time that appear in the output file records.

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Date Low and High</td>
<td>RecDateLo=</td>
<td>Determines the start and end date that appear in the output file records. You can select one of the following:</td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td>• System Date. The computer system date is used. This is the default.</td>
</tr>
<tr>
<td></td>
<td>&lt;field name&gt;</td>
<td>• One of the following date keywords. If you select a keyword in the Record Date Low box, you cannot select values in the Record Date High box or the Record Time boxes. These boxes are unavailable.</td>
</tr>
<tr>
<td></td>
<td>RNDATE</td>
<td>• Run Date (Today). The start and end date is the current day.</td>
</tr>
<tr>
<td></td>
<td>CURDAY</td>
<td>• Previous Day to Current Day. The start date is the previous day and the end date is the current day.</td>
</tr>
<tr>
<td></td>
<td>CURWEK</td>
<td>• Current Week/Month. 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.</td>
</tr>
<tr>
<td></td>
<td>CURMON</td>
<td>• Previous Day. The start and end date are the previous day.</td>
</tr>
<tr>
<td></td>
<td>PREDAY</td>
<td>• Previous Week/Month. 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.</td>
</tr>
<tr>
<td></td>
<td>PREWEK</td>
<td>• A date field (if defined in the Fields tab). The value in the date field is used as the date.</td>
</tr>
<tr>
<td></td>
<td>PREMON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RecDateHi=</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;field name&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-6  •  Date/Time Parameters
### Conversion Definition File Description

<table>
<thead>
<tr>
<th>ITUAM Conversion Builder Option</th>
<th>Definition File Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Time Low and High</td>
<td>RecTimeLo=</td>
<td>Determines the start and end time that appear in the output file records. Note that if a keyword is selected in the Record Date Low box, the Record Time boxes are unavailable.</td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;field name&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENTIRE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RecTimeHi=</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;field name&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENTIRE</td>
<td></td>
</tr>
</tbody>
</table>

You can select one of the following:

- System Time. The computer system time is used. This is the default.
- A time field (if defined in the Fields tab). The value in the time field is used as the time.
- Entire Day. Defines the start time as 00:00:00 and the end time as 23:59:59.

---

**Table 2-6 • Date/Time Parameters (Continued)**
**Shift Parameters**

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.

<table>
<thead>
<tr>
<th><strong>ITUAM Conversion Builder Option</strong></th>
<th><strong>Definition File Statement</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift Processing Enabled</td>
<td>ShiftsEnabled=</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>■ YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ NO</td>
<td></td>
</tr>
<tr>
<td>Shift Field</td>
<td>ShiftField=&lt;field&gt;</td>
<td>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 Date/Time Parameters on page B-19).</td>
</tr>
<tr>
<td>Shift Char</td>
<td>Shift&lt;day&gt;=</td>
<td>The number (1–9) that represents the shift, for example, 1 for the first shift, 2 for the second shift, etc.</td>
</tr>
<tr>
<td></td>
<td>DEFINE &lt;shift char&gt; &lt;end time&gt; [ &lt;shift char&gt; &lt;end time&gt; ...]</td>
<td></td>
</tr>
<tr>
<td>End Time</td>
<td></td>
<td>The time that the shift ends.</td>
</tr>
</tbody>
</table>

*Table 2-7  •  Shift Parameters*
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX  78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.
Trademarks

The following are trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2
DB2 Connect
IBM

Tivoli
z/OS

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.
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.

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.

conversion definition file • The conversion definition file defines the format of the usage metering file as well as the format of the output file to be produced by ITUAM Conversion Engine.

CPU • Acronym for central processing unit. The computational and control unit of a computer.

CSR File • The resource file that contains the data that is input into ITUAM. The records in this file are comma-delimited and can contain a very large number of resource identifiers and resources. See also identifier and rate code.

CSR+ File • CSR+ files are similar to CSR files, with the exception that the records in the CSR+ file contain an additional header at the beginning of the record.

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.

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 CSR or CSR+ 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.

**ITUAM Aggregation Engine** • ITUAM Aggregation Engine is a COM object that aggregates the records within the usage metering file by identifier values and produces a CSR file. Because the data in the usage metering file has been aggregated, the resulting CSR file requires less processing time. See also identifier.

**ITUAM Conversion Builder** • ITUAM Conversion Builder is a GUI application that you can use to create conversion definition files for the usage metering files. These definition files are fed into ITUAM Conversion Engine. See also ITUAM Conversion Engine and conversion definition file.

**ITUAM Conversion Engine** • ITUAM Conversion Engine is a COM object that enables usage metering files to be processed by ITUAM. ITUAM Conversion Engine reformats the data in the files into CSR files.

**ITUAM Job Runner** • ITUAM Job Runner is a console application that runs the data collection process. ITUAM Job Runner executes the jobs defined in a job file. Each job can run one or more data collectors.

**ITUAM Processing Engine** • ITUAM Processing Engine is composed of COM objects that process the CSR Files created by ITUAM Aggregation Engine or ITUAM Conversion Builder and load the output into the ITUAM database.

**.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.


**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 ITUAM Aggregation Engine
attributes
  Default element 2-89
  Job element 2-48 to 2-52
  Jobs element 2-44 to 2-46
  Process element 2-53 to 2-55
  Step element 2-57 to 2-60
  Steps element 2-56

C
CIMSAcct program
  about 2-10
  parameters for 2-65 to 2-69
CIMSAggregation.dll See ITUAM Aggregation Engine
CIMSBill program
  about 2-11
  parameters for 2-71 to 2-75
CIMSJob.xsd schema, about 2-6
CIMSLIB.wsf script, about 2-15
CIMSPrat program
  about 2-11
  files used by
    parameters file 2-18
    proration table 2-17
    parameters for 2-64
CIMSSort program
  about 2-11
  parameters for 2-70
CIMSSUtils.wsc script, about 2-15
CIMSWinDisk.xml file, structure of 4-6 to 4-9

CIMSWinProcess.wsf script, parameters for 3-27 to 3-28
Cleanup program
  about 2-12
  parameters for 2-84 to 2-85
COM objects
  CIMSAggregation.dll See ITUAM Aggregation Engine
Console program type, parameters for 2-86 to 2-87
conversion definition file components,
  description of B-2 to B-21
conversion scripts
  about 2-7
  parameters for
    CIMSWinProcess.wsf 3-27 to 3-28
    TDSz.wsf 5-17
    VMware.wsf 3-35 to 3-36
  standard parameters for 2-8

D
database
  defining for the data collection process 2-49
  program for loading 2-11
DB2 connect, installing and configuring for the
  TDSz data collector 5-2 to 5-3
DBLoad program
  about 2-11
  parameters for 2-76 to 2-77
Default element
  about 2-89
  attributes for 2-89
Defaults element, about 2-89
feed folder
  contents of 2-14
  creating 2-14

FileTransfer program
  about 2-12
  parameters for 2-79 to 2-84

ftp file transfer, attributes for 2-79 to 2-84

installing
  DB2 Connect for the TDSz data collector 5-2 to
  5-3

ITUAM Data Collectors 2-3
  Windows Process data collector on multiple
  servers 3-9 to 3-17
  manual installation 3-16 to 3-17
  remote installation 3-9 to 3-15

ITUAM 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

ITUAM Data Collectors
  installing 2-3
  running 2-120 to 2-122
  system architecture, described 2-4 to 2-15
  system specifications 2-3

ITUAM Integrator, using 2-90 to 2-118

ITUAM Job Runner
  about 2-4
  running 2-120 to 2-122

Job element
  about 2-48
  attributes for 2-48 to 2-52

job files
  about 2-5
  creating 2-27 to 2-89
  Default element
    about 2-89
    attributes for 2-89
  Defaults element, about 2-89
  Job element
    about 2-48
    attributes for 2-48 to 2-52
  Jobs element
    about 2-44
    attributes for 2-44 to 2-46
  Parameter element
    about 2-60
    CIMSAcct attributes 2-65 to 2-69
    CIMSBill attributes 2-71 to 2-75
    CIMSSort attributes 2-70
    Cleanup attributes 2-84 to 2-85
    Console program type attributes 2-86 to 2-87
    DBLoad attributes 2-76 to 2-77
    FileTransfer attributes 2-79 to 2-84
    Scan attributes 2-62 to 2-63
    WaitFile attributes 2-78 to 2-79
    Parameters element, about 2-60
  Process element
    about 2-53
    attributes for 2-53 to 2-55
  SampleDeployProcessCollector.xml sample file
  SampleNightly_TDS.xml sample file 2-5
  SampleNightly_xSeries.xml sample file 2-5
schema, about 2-6
Step element
  about 2-57
  attributes for 2-57 to 2-60
Steps element
  about 2-56
  attribute for 2-56
structure of 2-44 to 2-89
testing 2-9, 2-121
job log files
  about 2-9
  defining the type and content of 2-9
  e-mail distribution of 2-9
  return codes in 2-10
Jobs element
  about 2-44
  attributes for 2-44 to 2-46
K
  kernel and user mode, about 3-24
L
  log dates
    about 2-4 to 2-5
    passing
      default parameter 2-5
      from the command line 2-5, 2-121
      from the job file 2-5, 2-28, 2-89
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 2-84 to 2-85
    Console program type 2-86 to 2-87
    DBLoad program 2-76 to 2-77
    FileTransfer program 2-79 to 2-84
    Scan program 2-62 to 2-63
    WaitFile program 2-78 to 2-79
  Parameters element, about 2-60
  process definition folders
    about 2-13
    contents of 2-14
    creating 2-13
  Process element
    about 2-53
    attributes for 2-53 to 2-55
Processes folder
  creating 2-13
  path to, defining 2-13
  processing
    data
      permissions (folder) required for 2-119
      suggested frequency 2-119
    programs, descriptions of 2-10 to 2-12
  proration
    about 2-16
    example 2-23 to 2-26
    parameters file for
      creating 2-18 to 2-22
      example 2-24 to 2-25
    table
      creating 2-17
      example 2-23
R
  return codes, descriptions of 2-10
  running ITUAM Data Collectors 2-120 to 2-122
S
  SampleDeployProcessCollector job file, about 3-9
  SampleNightly_TDS.xml job file, about 2-5
  SampleNightly_xSeries.xml job file, about 2-5
  Scan program
    about 2-10
    parameters for 2-62 to 2-63
    Smart Scan feature
      enabling 2-40 to 2-43
      example of use in job file 2-40 to 2-43
  schema for job files, about 2-6
  Shell.wsc script, about 2-15
  SingleProcessStep program, about 2-11
  Smart Scan
    enabling 2-40 to 2-43
    example of use in job file 2-40 to 2-43
  Step element
    about 2-57
    attributes for 2-57 to 2-60
  Steps element
    about 2-56
    attribute for 2-56
system
  architecture, described 2-4 to 2-15
  specifications 2-3

T
TDSz data collector
  about 5-2
  conversion definition files setting up 5-4 to 5-13
  DB2 Connect, installing and configuring 5-2 to 5-3
  setting up 5-15 to 5-17
  TDSz tables supported 5-4
  transferring files, program for 2-12

U
UNIX data collector
  about 6-2
  job file process example 6-2
  setting up 6-2 to 6-4

V
variable values used for data collection 2-61
VMware data collector
  about 3-33
  identifiers and resources collected by 3-33
  job file process example 3-34
  setting up 3-34 to 3-36
  VMware.wsf script, parameters for 3-35 to 3-36

W
WaitFile program
  about 2-12
  parameters for 2-78 to 2-79
Windows Disk data collector
  about 4-2
  identifiers and resources collected by 4-2
  job file process example 4-3
  setting up 4-3 to 4-9
  Smart Scan, requirements for 4-9
  XML file, structure of 4-6 to 4-9
Windows disk storage data collector See Windows Disk data collector
Windows operating system data collector See Windows Process data collector

Windows Process data collector
  about 3-2
  installing on multiple servers 3-9 to 3-17
    manual installation 3-16 to 3-17
    remote installation 3-9 to 3-15
  job file process example
    on central ITUAM Data Collectors Server 3-26
    on other server 3-31
  kernel and user mode, about 3-24
  log files
    enabling the creation of 3-18 to 3-20
    format of 3-21 to 3-23
    identifiers and resources collected from 3-25
  service log on user account, creating 3-2
  setting up 3-26 to 3-32
  starting 3-20
  system configuration options 3-4 to 3-8