IBM SmartCloud Cost Management

Version 2.1.0.5

User's Guide



IBM SmartCloud Cost Management

Version 2.1.0.5

User's Guide



Note
Before you use this information and the product it supports, read the information in Notices.

This edition applies to version 2, release 5, fix pack 1 of IBM Cloud Orchestrator (program number 5725-H28) and to all subsequent releases and modifications until otherwise indicated in new editions.

The material in this document is an excerpt from the IBM Cloud Orchestrator knowledge center and is provided for convenience. This document should be used in conjunction with the knowledge center.

© Copyright IBM Corporation 2013, 2015.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface .. xi
Who should read this information .. xi

Chapter 1. Installing SmartCloud Cost Management 2.1.0.5. .. 1
Installation overview .. 1
Supported hardware and software requirements .. 1
Requirements for Linux servers .. 2
Installation dependencies .. 2
Installing the downloaded software .. 3
Installing using the installation script .. 3
Uninstalling SmartCloud Cost Management .. 4
Upgrading from SmartCloud Cost Management 2.1.0.4. .. 4

Chapter 2. Configuration required for metering .. 7
Automated configuration .. 7
Logging in to the Administration Console .. 8
Accepting the security certificate .. 9
Configuring the JDBC Driver .. 9
Adding a JDBC driver .. 9
SmartCloud Cost Management Command Line Interface .. 10
Using the Command Line Interface to manage data sources .. 11
Using the Command Line Interface to manage load tracking .. 12
Configuring the SmartCloud Cost Management data sources .. 13
Adding a Database data source .. 14
Setting the Default Admin and Processing data source .. 16
Adding a Web service data source .. 16
Adding a Server data source .. 17
Adding a Message Broker data source .. 18
Maintaining data sources .. 19
About initializing the database .. 19
Initializing the database .. 20
Loading the database with sample IBM Cloud Orchestrator data .. 20
Security overview .. 21
Account Code security for IBM Cloud Orchestrator .. 21
IBM Cloud Orchestrator Account code structure .. 22
User management .. 23
Configuring Keystone as a Central User Registry .. 23
Defining Tivoli Common Reporting security permissions .. 24
Managing Jazz for Service Management roles for users .. 24
Configuring security permissions .. 25
Constraining access to reports .. 26

Chapter 3. Administering the system 27
Defining offerings and rate groups .. 27
Adding offerings .. 27
Deleting offerings .. 28
Adding rate groups. .. 28
Changing the rate group sequence .. 29
Deleting rate groups .. 29
Defining rates using the Rate Tables panel .. 29
Adding rate tables .. 30
Removing rate tables .. 30
Defining rates .. 31
Viewing rates .. 31
Adding rates .. 32
Adding rates with tiers .. 33
Adding rates with shifts .. 35
Importing rates .. 36
Importing rates from an existing rate table 36
Moving rates between rate groups .. 38
Changing the rate sequence .. 38
Modifying rates .. 39
Deleting rates and rate group rates .. 42
Defining rate templates .. 42
Resource elements for the rate template .. 43
Modifying the sample rate templates .. 45
Importing the rate template .. 46
Deleting the rate template .. 47
Defining the calendar .. 47
Calendar considerations .. 47
Setting up the calendar .. 47
Defining configuration options .. 49
Adding a JDBC driver .. 49
Setting logging options .. 50
Adding organization information .. 50
Setting processing options .. 51
Setting reporting options .. 52
Stopping and starting the Application Server .. 52

Chapter 4. Administering data processing .. 53
Data processing architecture .. 53
Job Runner .. 53
Job files. .. 53
Job file XML schema .. 54
Collection files .. 54
Log files .. 55
Message and trace log files .. 55
Job log files .. 55
Embedded Web Application Server log files 56
Installation log files. .. 57
Processing programs .. 57
Process definitions .. 58
Date keywords .. 59
Data processing overview .. 60
Data processing frequency .. 60

© Copyright IBM Corp. 2013, 2015

Required directory permissions for data processing 61
SmartCloud Cost Management Processing Engine 61
Acct 62
Bill 63
DBLoad 65
ReBill 66
SmartCloud Cost Management Integrator . . 68
Setting up and running job files 69
Creating job files 69
Job file structure 70
Jobs element 70
Job Element 72
Process element 74
Steps Element 76
Step Element 76
Parameters Element 79
Parameter element 79
Acct specific parameter attributes . . . 80
Bill specific parameter attributes . . . 82
Cleanup specific parameter attributes. . 84
Console parameter attributes 84
Console specific parameter attributes . . 85
DBLoad specific parameter attributes . . 86
DBPurge specific parameter attributes . . 88
FileTransfer specific parameter attributes . 88
Rebill specific parameter attributes . . 92
Scan specific parameter attributes . . . 93
WaitFile specific parameter attributes . . 94
Defaults Element 95
Default Element 95
Integrator job file structure 97
Input element 97
Stage elements 100
Running job files 161
Running job files in batch mode 162
Viewing job file logs 163
Viewing sample job files. 164
Setting accounting dates. 164
Account codes and account code conversion . . 164
Setting up account codes and performing account code conversion. 165
Defining the account code structure . . . 165
Defining the account code 165
Defining the account code input field using the Integrator program 166
Defining the account code input field using the Acct program 166
Converting account code input field values to uppercase. 167
Setting the account code conversion options (Acct program) 167
Enabling account code conversion . . 168
Defining the identifier values for account code conversion 168
Defining optional move fields 168
Defining the account code conversion table 169
Enabling exception processing. . . . 170
Creating the account code conversion table (Acct program) 170
Account code conversion example (Acct program) 172
Advanced account code conversion example (Acct program) 173
Setting the account code conversion options (Integrator program) 176
Enabling account code conversion . . 176
Defining the identifier values for account code conversion 176
Defining optional move fields 177
Defining the account code conversion table 179
Enabling exception processing. . . . 179
Creating the account code conversion table (Integrator program) 180
Account code conversion example (Integrator program) 181
Setting up conversion mappings 183
Adding conversion mappings 183
About exception file processing 183
Setting up shifts 183
Transferring files 184
Sample FTP file transfer job file 186
Sample SSH file transfer job file 187
Sample local file transfer job file 188
Using an encrypted password to connect to a remote system 190
File encoding 190
Default file encoding 191
Overriding default file encoding 191

Chapter 5. Administering reports . .. 193

Working with Cognos based Tivoli Common Reporting 193
About the Tivoli Common Reporting application 193
Running reports 195
Using the Cognos toolbar 195
Printing reports 196
Saving reports 196
Report properties 196
Creating Dashboard reports 196
Creating a sample dashboard report . . . 197
Creating Page dashboard reports 198
Working with custom reports 199
Creating, editing, and saving custom reports 199
Creating or updating a custom report .. 199
Saving a custom report 200
Using report parameters. 200
Simple parameters. 201
Existing parameters 202
Using report filters 204
Template reports 206
Template Account Code Date 207
Template Account Code Year 207
Rebranding reports 208
Working with the SmartCloud Cost Management Cognos model 209
Accessing the Cognos model 209
Adding custom objects to the Cognos model 209
Publishing the custom model 210

Additional resources 211

Chapter 6. Administering the database 213

Tracking database loads 213
Administering database objects 214
Administering database tables 215
Loading table dependencies 216
DataAccessManager command-line utility . . . 218
DDL extraction 219
Populate tables 220
Setting the database version 220
Importing data into a database table. 221
Exporting data into a database table 222

Chapter 7. Administering data collectors 223

Universal Collector overview 223
Creating a new collector using XML. 223
Creating new collectors using Java 234
IBM SmartCloud Cost Management Core data collectors 236
AIX Advanced Accounting data collector . . . 236
Process data collected (Advanced Accounting record type 1) 236
System data collected (Advanced Accounting record type 4) 237
File system data collected (Advanced Accounting record type 6) 238
Network data collected (Advanced Accounting record type 7) 238
Disk data collected (Advanced Accounting record type 8) 239
Virtual I/O server data collected (Advanced Accounting record type 10). 239
Virtual I/O client data collected (Advanced Accounting record type 11). 240
ARM transaction data collected (Advanced Accounting record type 16). 240
WPAR system data collected (Advanced Accounting record type 36). 241
WPAR file system data collected (Advanced Accounting record type 38). 241
WPAR disk data collected (Advanced Accounting record type 39). 242
Setting up AIX Advanced Accounting data collection 243
HPVMSar data collector 243
Identifiers and resources collected by hpvmsar collector 243
Deploying the SmartCloud Cost Management hpvmsar collector 244
Installing the SmartCloud Cost Management hpvmsar Collector from the SmartCloud Cost Management Server .. 244
Manually installing the hpvmsar collector 245
Following the installation of the hpvmsar collector 246
Uninstalling the hpvmsar collector . .. 246
OpenStack data collector 247
Configuring the OpenStack data collector .. 247
Manually configuring the OpenStack data collector 247
OpenStack identifiers and resources 254
Compute identifiers and resources . .. 254
Volume identifiers and resources 257
OpenStack job file 260
Context job file 260
VM Instances job file 264
Volumes job file 268
IBM PowerVM HMC data collector 270
Configuring HMC to enable SmartCloud Cost Management HMC collection 270
HMC collector log file format 270
Identifiers and resources defined by HMC collector 275
Setting up SmartCloud Cost Management for HMC data collection 278
HMC data commands 281
KVM data collector 282
Identifiers and resources collected by the KVM collector 283
Deploying the SmartCloud Cost Management KVM collector 284
Installing the SmartCloud Cost Management KVM collector from the SmartCloud Cost Management Server .. 284
Manually installing the SmartCloud Cost Management KVM collector 285
Following the installation of the SmartCloud Cost Management KVM collector 286
Uninstalling the SmartCloud Cost Management KVM collector 286
Linux, z/Linux, UNIX, and AIX operating system and file system data collectors 287
Sar data collector 287
Identifiers and resources collected by sar collector 287
Deploying the sar collector 288
Installing the SmartCloud Cost Management sar Collector from the SmartCloud Cost Management Server .. 289
Manually installing the SmartCloud Cost Management sar collector 290
Following the installation of the sar collector 290
Uninstalling the sar collector 291
Tivoli Data Warehouse data collector 291
Setting up Tivoli Data Warehouse data collection 291
Configuring the Tivoli Data Warehouse job file 291
Active Energy Manager identifiers and resources 295
AIX Premium identifiers and resources . .. 298
Eaton identifiers and resources 305
HMC identifiers and resources 307
ITCAM/SOA identifiers and resources . .. 308
Linux OS identifiers and resources 310
UNIX OS identifiers and resources 312
Windows OS identifiers and resources . .. 315

Universal data collector 326
Setting up the Universal data collector . . 326
Virtual I/O Server data collector 326
Identifiers and resources defined by the Virtual I/O Server data collector 326
Transferring usage logs from the Virtual I/O Server to the SmartCloud Cost Management application server for processing 329
Setting up Virtual I/O Server data collection 333
VMware data collector 333
Configuring VMware server systems to enable SmartCloud Cost Management VMware collection 333
VMware usage metrics collected 334
Identifiers and resources defined by the VMware collector 338
Setting up SmartCloud Cost Management for VMware data collection 340
Mapping of VMware Identifiers and resources to VMware Infrastructure Equivalents 343
Vmstat data collector 344
Identifiers and resources collected by vmstat collector 344
Deploying the vmstat collector 345
Installing the SmartCloud Cost Management vmstat Collector from the SmartCloud Cost Management Server .. 345
Manually installing the vmstat collector 346
Following the installation of the vmstat collector 347
Uninstalling the vmstat collector . . . 348
Windows Disk data collector 348
Identifiers and resources collected by the Windows Disk collector 348
Setting up Windows Disk data collection .. 350
File system collection 354
Physical and logical disk collection . . . 354
Windows Process data collector 356
Creating a log on user account for the Windows Process collector service (optional) . 356
Assigning Polices at the domain level .. 357
Assigning Polices at the local level . . 357
System configuration options for the Windows Process collector 358
Installing the Windows Process collector .. 359
Installing remotely 359
Installing manually 362
Windows Process collector log file format 364
Identifiers and resources defined by the Windows Process collector 368
Setting up Windows Process data collection 370
Troubleshooting 373
z/VM data collector 374
z/VM standard billable items collected . . 374
z/VM Accounting Records Processed . . 374
Creating a process definition directory . . 375
Installing the z/VM collector 376
Binding the CIMSCMS program . . . 376
Running the z/VM Collector 376
CIMSCMS Control Statements 378
About the Z/VM collector output CSR file 381
Transferring output CSR files from the z/VM system 382
Setting up z/VM data collection 382
Setting up for data collection on a Linux or UNIX system 382
Installing SmartCloud Cost Management Data Collectors for UNIX and Linux 382
Linux and UNIX data collection architecture .. 387
Setting the sharable library path 389
Setting the environment variables for data collection and consolidation 389
Setting up operating and file system data collection: starting Linux or UNIX process accounting 394
Identifiers and resources defined from process accounting data (operating system and file system) 395
Setting up AIX Advanced Accounting data collection 397
Installing the UNIX/Linux AIX Advanced Accounting Collector 398
Installing the UNIX/Linux AIX Advanced Accounting Collector Manually . . . 398
Installing the UNIX/Linux AIX Advanced Accounting Collector using RXA deployment 398
Post Install Configuration 398
Creating Advanced Accounting data files .. 399
Configuring AIX Advanced Accounting .. 399
Setting up Advanced Accounting data collection 400
Scheduling Advanced Accounting data collection 402
Advanced Accounting metrics collected and SmartCloud Cost Management rate codes .. 402
Setting up Virtual I/O Server data collection 406
About the Configuration Parameter file on the Virtual I/O Server 406
Configuring and starting the SmartCloud Cost Management Agent 407
Collecting and Converting Virtual I/O Server data files 408
Transferring Virtual I/O Server usage logs to the SmartCloud Cost Management application server 409
Virtual I/O Server metrics collected and SmartCloud Cost Management rate codes . 411
Schedule the data collection and consolidation scripts 413
Collecting data: setting up the data collection scripts 414
Check pacct File Script (check_pacct) . . 415
Nightly Accounting Script (ituam_uc_nightly) 415
Turn Accounting Script (turnacct). 416
Run Account Script (runacct) 416
Sampler Script (sampler) 417
Database Storage Scripts (get_odb_storage and get_db2_storage) 418
Redo Nightly Script (redo_nightly) . . . 418

Consolidating data: setting up the data consolidation scripts 419
Nightly Consolidation Script (CS_nightly_consolidation) 419
Running a UNIX script file from within a job file 419
CSR file types 421
Transferring CSR files to the SmartCloud Cost Management application server 422
Transferring log files to the SmartCloud Cost Management application server 425

Chapter 8. Troubleshooting 427
Troubleshooting information 427
Common problems and solutions. 427
Troubleshooting installation 427
Manually uninstalling 427
Manually uninstalling SmartCloud Cost Management 428
Manually uninstalling Windows Process Collector 428
Unable to connect to the Administration Console 429
Troubleshooting administration 429
Server instance ports in use 429
Active Reports do not open or are incorrectly displayed when rendered in Japanese, Chinese, or Korean 430
Administration Console runs slowly, hangs, or will not connect to the database 430
Unable to run Tivoli Common Reporting reports that use account code prompts . . . 430
Report data not updated when running reports 431
Issues when using the SmartCloud Cost Management rate template 432
Troubleshooting database applications . . . 432
Troubleshooting for Exception: DB2 SQL error: SQLCODE: -964, SQLSTATE: 57011, SQLERRMC: null 432
Stored Procedure performance issues on DB2 433
Using log files 434
Message and trace log files 434
Job log files 434
Embedded Web Application Server log files . . 435
Installation log files 436
Getting SmartCloud Cost Management system information 436
Disabling Internet Explorer Enhanced Security Configuration 436
Resolving the `FileNotFound Exception` error on UNIX and Linux systems 437

Chapter 9. Reference 439
Encryption information 439
FIPS Compliance 439
Enabling FIPS on the Application Server . . . 439
Enabling Tivoli Common Reporting for FIPS 440
Setting the SmartCloud Cost Management processing path. 440
REST API reference 440
Symbols and abbreviated terms 441
REST API reference overview 441
Operations 443
Protocol resource elements 445
Using Apache Wink to access SmartCloud Cost Management REST Resources 447
Clients Rest API 449
GET Resource clients 449
GET{id} Resource clients. 451
POST Resource clients 452
PUT Resource clients 453
DELETE Resource clients 454
Users REST API 455
GET Resource users 455
GET{id} Resource users 456
POST Resource users 457
PUT Resource users 458
DELETE Resource users 460
Usergroups REST API 460
GET Resource usergroups 460
Get{id} Resource usergroups 462
POST Resource usergroups 464
PUT Resource usergroups 466
DELETE Resource usergroups 469
AccountCodeStructrues REST API 470
GET Resource accountCodeStructrues . . . 470
GET{id} Resource accountCodeStructrues . . 472
POST Resource accountCodeStructrues . . . 474
PUT Resource accountCodeStructrue . . . 476
DELETE Resource accountCodeStructrues 479
Job file structure 480
Jobs element. 480
Job Element 483
Process element 485
Steps Element 486
Step Element 487
Parameters Element 490
Parameter element. 490
Acct specific parameter attributes. 491
Bill specific parameter attributes 493
Cleanup specific parameter attributes . . . 495
Console parameter attributes 495
Console specific parameter attributes . . . 496
DBLoad specific parameter attributes . . . 497
DBPurge specific parameter attributes . . . 499
FileTransfer specific parameter attributes . . 499
Rebill specific parameter attributes 503
Scan specific parameter attributes 504
WaitFile specific parameter attributes . . . 505
Defaults Element 506
Default Element 506
Integrator job file structure 508
Input element 508
Stage elements 511
Aggregator 511
CreateAccountRelationship 514
CreateIdentifierFromIdentifiers 517
CreateIdentifierFromRegEx 519
CreateIdentifierFromTable 520
CreateIdentifierFromValue 525

CreateResourceFromConversion .. 526
CreateResourceFromDuration .. 529
CreateResourceFromValue .. 532
CreateUserRelationship .. 533
CSROutput .. 535
CSRPlusOutput. .. 535
DropFields .. 536
DropIdentifiers .. 537
DropResources .. 538
ExcludeRecsByDate .. 539
ExcludeRecsByPresence .. 540
ExcludeRecsByValue .. 541
FormatDateIdentifier .. 543
IdentifierConversionFromTable .. 544
IncludeRecsByDate .. 550
IncludeRecsByPresence .. 551
IncludeRecsByValue .. 553
MaxRecords .. 555
PadIdentifier .. 556
Prorate .. 557
RenameFields .. 558
RenameResourceFromIdentifier .. 559
ResourceConversion .. 561
Sort .. 563
TierResources .. 564
TierSingleResource .. 567
UpdateConversionFromRecord .. 569
Control statements .. 572
Acct Program Control Statements. .. 573
ACCOUNT CODE CONVERSION {SORT} 573
ACCOUNT FIELD .. 573
DATE SELECTION .. 574
DEFINE FIELD. .. 575
DEFINE MOVEFLD .. 576
EXCEPTION FILE PROCESSING ON .. 576
PRINT ACCOUNT NO-MATCH .. 577
SHIFT .. 577
UPPERCASE ACCOUNT FIELDS .. 578
Bill Program Control Statements .. 578
BACKLOAD DATA .. 579
CLIENT SEARCH ON .. 579
DATE SELECTION .. 580
DEFAULT CLOSE DAY .. 581
DEFINE .. 582
DYNAMIC CLIENT ADD ON. .. 583
EXCLUDE .. 584
INCLUDE .. 585
KEEP ORIGINAL CPU VALUES .. 585
NORMALIZE CPU VALUES .. 586
REPORT DATE. .. 586
USE SHIFT CODES .. 588
Reports .. 588
Cognos based Tivoli Common Reporting reports 589
Account reports .. 590
Account Summary YTD report .. 590
Account Total Invoice report .. 591
Application Cost report .. 592
Daily Charges - Charges report .. 594
Daily Crosstab - Usage report .. 595
Monthly Crosstab - Charges report .. 597
Monthly Crosstab - Usage report .. 598
Percentage report .. 599
Summary Crosstab - Charges report .. 600
Summary Crosstab - Usage report .. 602
Weekly Crosstab - Charges report .. 603
Weekly Crosstab - Usage report .. 604
Budget reports .. 605
Client Budget report .. 605
Line Item Budget report .. 606
Cloud reports .. 607
Project Summary report .. 607
Dashboard reports. .. 610
Top N Account Charges report .. 610
Top N Account Charges Pie Chart report 611
Top N Rate Group and Rate Resource Usage report .. 612
Invoice reports .. 613
Invoice by Account Level report .. 613
Invoice Detail Line Item Resource Units by Identifiers report .. 616
Invoice Drill Down for Rate Group by Date report .. 616
Run Total Invoice report. .. 617
Run Total Rate Group Percent report .. 619
Other reports .. 620
Client report. .. 620
Configuration report .. 621
Rate report .. 621
Resource Detail reports .. 622
Batch report .. 622
Charges by Identifier report .. 623
Detail by Identifier report .. 626
Detail by Multiple Identifiers report .. 627
Usage by Identifier report .. 628
Template reports .. 631
Template report .. 631
Template Account Code Date report .. 631
Template Account Code Year report .. 632
Top Usage reports .. 633
Top 10 Bar Graph report. .. 633
Top 10 Cost report. .. 634
Top 10 Pie Chart report .. 636
Trend reports .. 637
Cost Trend report .. 637
Cost Trend by Rate report .. 638
Cost Trend Graph report .. 639
Resource Usage Trend report .. 640
Usage Trend Graph report .. 642
Variance reports .. 643
Cost Variance report .. 643
Resource Variance report .. 644
Tables .. 645
CIMSAccountCodeConversion Table. .. 645
CIMSCalendar Table .. 646
CIMSClient Table .. 646
CIMSClientBudget Table. .. 646
CIMSClientContact Table .. 647
CIMSClientContactNumber Table. .. 648
CIMSConfig Table .. 648
CIMSConfigAccountLevel Table .. 649
CIMSConfigOptions Table .. 649
CIMSCPUNormalization Table .. 650

CIMSDetailIdent Table 650
CIMSDIMClient Table 651
CIMSDIMClientContact Table 651
CIMSDIMClientContactNumber Table 652
CIMSDIMUserRestrict Table 653
CIMSIdent Table 653
CIMSLoadTracking Table 654
CIMSProcDefinitionMapping Table 654
CIMSProcessDefinition Table 655
CIMSProrateSource Table 655
CIMSProrateTarget Table 656
CIMSRateGroup Table 656
CIMSRateIdentifiers Table 657
CIMSReportCustomFields Table 657
CIMSReportDistribution Table 657
CIMSReportDistributionParm Table 658
CIMSReportDistributionType Table 658
CIMSReportGroup Table 658
CIMSReportToReportGroup Table 659
CIMSSummaryToDetail Table 659
CIMSTransaction Table 659
CIMSUser Table 661
CIMSUserConfigOptions Table 661
CIMSUserGroupAccountCode Table 662
CIMSUserGroupAccountStructure Table . . 662
CIMSUserGroupConfigOptions Table 662
CIMSUserGroupReport Table 663
CIMSUserGroup Table 663
CIMSUserGroupProcessDef Table 664
CIMSUsertoUserGroup Table 664
SCDetail Table 665
SCDIMCalendar Table 666
SCDIMRate Table 666
SCDIMRateShift Table 671
SCRate Table 676
SCRateTable Table 678
SCRateShift Table 679
SCResourceUtilization Table 679
SCSummaryDaily Table 680
SCSummary Table 680
SCUnits Table 682
SmartCloud Cost Management files 683
CSR file 683
CSR+ file 685
Ident file 688
Resource file. 688
Detail file. 689
Summary file 691

Accessibility features for SmartCloud Cost Management 693

Notices 695
Programming interface information 697
Trademarks 697
Terms and conditions for product documentation 697
IBM Online Privacy Statement. 698

Glossary 699
A 699
B 700
C 700
E 700
H 700
K 700
P 701
R 702
S 703
T 703
V 704

Preface

This publication documents how to use IBM® SmartCloud Cost Management..

Who should read this information

This information is intended for administrators who install and configure IBM SmartCloud Cost Management, and for users who work with this product.

© Copyright IBM Corp. 2013, 2015

Chapter 1. Installing SmartCloud Cost Management 2.1.0.5

Use the procedures in this section to install SmartCloud Cost Management components on a Linux operating system .

Installation overview

This section describes the installation and configuration options available in SmartCloud Cost Management.

SmartCloud Cost Management uses the following for reporting:

- Cognos® based Tivoli® Common Reporting
  - A list of the Cognos reports is available in the related *Reference Reports* section.
  - Refer to the related links when installing Tivoli Common Reporting 3.1.0.1.

Installation options

The following install method is available when installing SmartCloud Cost Management products on Linux operating systems:

- Download the installation images from the IBM Passport Advantage® site, if you are licensed to do so.

Use the provided installation script to perform the installation.

Note: To download SmartCloud Cost Management from Passport Advantage®, see the download instructions on the Passport Advantage website.

Installation configuration

The standard configuration for SmartCloud Cost Management is a distributed installation with SmartCloud Cost Management, Tivoli Common Reporting, and DB2 database each on their own system. The DB2 database is shared with other OpenStack components. The Jazz™ for Service Management server that is used to host Tivoli Common Reporting can also be shared with other components of IBM SmartCloud® Cloud Orchestrator.

Supported hardware and software requirements

Before you install the SmartCloud Cost Management application server, review the software and hardware requirements for Linux platforms.

Related reference:

"Installation dependencies" on page 2
Make sure that you understand the dependencies outlined in this topic before you install SmartCloud Cost Management.

© Copyright IBM Corp. 2013, 2015

Requirements for Linux servers

This topic describes prerequisites for installing SmartCloud Cost Management on a Linux operating system.

Linux platforms

Note: For the complete listing of software product compatibility reports, see Software product compatibility reports.

Software and hardware	Requirements
Operating system	• Red Hat Enterprise Linux (RHEL) 6.4, 6.5, 7.0, and 7.1 for x64 (AMD 64). **Note**: RHEL 6.4 and 6.5 is supported by Tivoli Common Reporting 3.1.0.1. • SUSE Linux Enterprise Server (SLES) 10 and 11 for x64 (AMD64).
Browser	• Mozilla Firefox ESR31, ESR38 • Internet Explorer 11 • Chrome 45
Hard disk drive space	5 GB minimum, 40 GB recommended (available hard disk drive space) **Note:** The hard disk drive space requirements for your organization might vary.
Processor speed	3 GHz minimum
Memory	2 GB minimum

Installation dependencies

Make sure that you understand the dependencies outlined in this topic before you install SmartCloud Cost Management.

Ensure that the software and hardware requirements have been satisfied

Make sure that you have your system setup correctly. For the latest updates on hardware and software requirements, see the related reference topic.

The user who is installing SmartCloud Cost Management has the required privileges

SmartCloud Cost Management can be installed as root or a non-root user. If installing as a non-root user, ensure the user has permissions to write to the directory you want to install into, and that you do not specify any port numbers less than 1024.

Related concepts:

"Supported hardware and software requirements" on page 1
Before you install the SmartCloud Cost Management application server, review the software and hardware requirements for Linux platforms.

Installing the downloaded software

This topic describes how to install when downloading respective files from IBM Passport Advantage®.

Before you begin

Before you install, download and extract the software.

About this task

Download the SmartCloud Cost Management image from the IBM Passport Advantage® site. The downloaded file name is:

`SCCM_2105_Linux_x64.tar`

Procedure

Extract the image file to intended installation system disk, using the following command:`tar -xf SCCM_2105_Linux_x64.tar -C /tmp/sccm_install`

Results

The product is ready to install using the installation script.

Installing using the installation script

The installation for SmartCloud Cost Management is provided as a console mode install only. This may be used regardless of whether a GUI environment is available or not.

About this task

Installation of SmartCloud Cost Management 2.1.0.5 requires you to choose an installation directory, and optionally to choose the http and https ports that the server will be available on. If not specified, the ports will default to `9080` and `9443` respectively.

Procedure

1. Log in to the system using the login credentials required for installing on a Linux platform .
2. Enter one of the following commands:

   ```
   ./sccm_install.sh /opt/ibm/sccm
   ```

   or

   ```
   ./sccm_install.sh sccm_install.properties
   ```

   Note: The `sccm_install.properties` file can be modified as required.
3. Follow the directions presented on the screen to complete the installation.
4. If the installation was run as a non-root user, the SmartCloud Cost Management Application Server is not automatically configured to run on system boot, as the installer would not have had permissions to do so. If you want to manually configure this autostart, run the following script as root:

   `SCCM_install_dir/bin/configure_autostart.sh`

5. If the installation is run as a non-root user, Network Time Protocol will not have been configured to sync the system time to the IBM Cloud Orchestrator system. For accurate metering data, ensure that the system clock is synchronized with the IBM Cloud Orchestrator system by copying the `/etc/ntp.conf` file from one of the IBM Cloud Orchestrator systems to the system running SmartCloud Cost Management, then restart the ntp service using:

   ```
   service ntp restart
   ```

6. Launch the browser: `https://<host>:<port>/Blaze/Console`. For example, `https://<servername>:9443/Blaze/Console`

Uninstalling SmartCloud Cost Management

If required, you can uninstall SmartCloud Cost Management using the steps described in this section.

Before you begin

Ensure you back up any files you need to keep before running the uninstall script.

About this task

The installation process places an `uninstall.sh` script in the `SCCM_install_dir` directory.

Procedure

1. Go to the `SCCM_install_dir` directory and run the `uninstall.sh` script as the same user that was used to do the install, or as the root user.
2. If you ran the script as a non-root user, you must remove the automatic startup scripts from SmartCloud Cost Management. To do this, execute the following commands as root user:

   ```
   chkconfig --del ibm-sccm
   rm /etc/init.d/ibm-sccm
   ```

Results

The script stops the SmartCloud Cost Management and Metering Control Service servers and removes the software, along with any configuration or other items under SCCM_install_dir.

Note: If required, you can also do a manual uninstallation of the product. See related topic for more information.

Upgrading from SmartCloud Cost Management 2.1.0.4

You can upgrade from SmartCloud Cost Management 2.1.0.4 to SmartCloud Cost Management 2.1.0.5 by using the `sccm_install.sh` script. The upgrade is provided as a console mode upgrade only.

Before you begin

If you are using SmartCloud Cost Management for metering OpenStack and you are upgrading to IBM Cloud Orchestrator 2.5.0.1, follow the procedure in Preparing for the migration.

Ensure that IBM Cloud Orchestrator is successfully upgraded to IBM Cloud Orchestrator 2.5 Fix Pack 1.

Ensure that the system that you are upgrading is the same system where SmartCloud Cost Management 2.1.0.4 has been installed. The properties defined in the `sccm_install.properties` file during the installation of SmartCloud Cost Management 2.1.0.4 must also be used for the 2.1.0.5 upgrade. These properties include:

INSTALL_DIR
: The installation directory of SmartCloud Cost Management. This directory must be the same as that used in 2.1.0.4 because the `sccm_install.sh` script uses this value to detect if there is a previous installation and therefore if it needs to run in upgrade mode.

ADMIN_USER
: The SmartCloud Cost Management administration user.

ADMIN_PASSWORD
: The SmartCloud Cost Management administration password.

HTTPS_PORT
: The https port which should be used by SmartCloud Cost Management.

Procedure

1. Log in to the system using the login credentials required for installing on a Linux platform .
2. Enter one of the following commands:
   - `./sccm_install.sh`
   - `./sccm_install.sh sccm_install.properties`

   Note: The `sccm_install.properties` file can be modified as required but must match the parameters used in the 2.1.0.4 installation.
3. Follow the directions presented on the screen to complete the installation.
4. Launch the browser: `https://<host>:<port>/Blaze/Console`. For example, `https://<servername>:9443/Blaze/Console`.
5. Once installation is successfully completed, run the automated post-installation configuration to ensure that SmartCloud Cost Management is configured to work with IBM Cloud Orchestrator 2.5 Fix Pack 1.

   Note: Metering and Billing will not work correctly for an upgraded IBM Cloud Orchestrator 2.5 Fix Pack 1 environment unless configured correctly to do so.

Chapter 2. Configuration required for metering

This section describes the configuration tasks required before using SmartCloud Cost Management for metering.

Automated configuration

Most of the post-installation configuration of SmartCloud Cost Management for IBM Cloud Orchestrator is automated. This automation process is controlled by running the `ico_configure.sh` script as the same user that installed SmartCloud Cost Management.

Before running the `ico_configure.sh` script, you must ensure that the following conditions are satisfied:

- IBM Cloud Orchestrator is installed and configured. For more information, refer to the related installation section.
- The IBM Cloud Orchestrator Server is registered in DNS and is resolvable.
- Jazz for Service Management 1.1.0.1 is installed with Tivoli Common Reporting 3.1.0.1.

You can use the `ico_configure.sh` script during initial configuration and whenever the password of the Keystone admin users changes or new regions are added or migrated from a previous version of IBM Cloud Orchestrator.

Run the `ico_configure.sh` script as follows:

```
cd <SCCM_install_dir>/bin/postconfig
./ico_configure.sh --ico-server <ICO_server> --sccmuser <smadmin> --sccmpass <smpass> \
   --jazz <jazzsmhost> --jazzuser <smadmin> --jazzpass <smpass>
```

where:

<ICO_server>
: Is the host name of the IBM Cloud Orchestrator Server.

<smadmin>
: Is the default admin user name for the SmartCloud Cost Management or Jazz for Service Management Administration Console.

<smpass>
: Is the password for the default admin user for the SmartCloud Cost Management or Jazz for Service Management Administration Console.

<jazzsmhost>
: Is the host name of the Jazz for Service Management server.

You may be prompted for the root passwords to the IBM Cloud Orchestrator Server and the Jazz for Service Management server during execution of this script.

The script automatically completes the following configuration steps if the `--ico-server` parameter is specified:

- Creates the SmartCloud Cost Management DB2 database on the IBM Cloud Orchestrator Server.
- Configures the `ico_db2` data source in SmartCloud Cost Management and initializes the database.

© Copyright IBM Corp. 2013, 2015

- Configures the `os_keystone` data source in SmartCloud Cost Management and runs the `OpenStackContext.xml` job file.
- Sets up a cronjob to automatically run the various OpenStack job files to collect context and metering data from IBM Cloud Orchestrator.
- Configures the Central User Registry to allow Keystone admin users to log in to SmartCloud Cost Management Administration Console.
- Enables metering notifications for all registered regions.
- Configures the data sources for all registered IBM Cloud Orchestrator regions in SmartCloud Cost Management.
- Enables SmartCloud Cost Management to listen to metering events from IBM Cloud Orchestrator.
- Imports the offering templates for IBM Cloud Orchestrator which are used to create default rate codes and values.

The script automatically completes the following configuration steps if the Jazz-related parameters are specified:

- Imports the offering templates for IBM Cloud Orchestrator which are used to create default rate codes and values.
- Imports the SmartCloud Cost Management reporting package into Jazz for Service Management reporting.
- Configures the SmartCloud Cost Management data source in Jazz for Service Management reporting.
- Assigns all Jazz for Service Management users access to run the reports.
- Restricts Tivoli Common Reporting Administration and Authoring capabilities to the Jazz for Service Management Admin user.

Note: Running this script to configure Tivoli Common Reporting secures the Authoring (Report Studio) and Administration functions with Tivoli Common Reporting. This may affect other products installed that use Tivoli Common Reporting. You must check that the security access in Tivoli Common Reporting is appropriate for all products after installation.

Related reference:

"Configuring the connection to OpenStack" on page 249
The Metering Control Service (MCS) uses an Advanced Message Queuing Protocol (AMQP) listener to collect notification events, such as Nova Compute notifications from the OpenStack RabbitMQ or Apache Qpid message broker.

Logging in to the Administration Console

Log in to the Administration Console to set up and configure the SmartCloud Cost Management.

About this task

To log in to the Administration Console:

Procedure

1. Start an Internet Explorer or Firefox Web browser, and type `https://<hostname>:9443/Blaze/Console/` in the address bar. In this case `<hostname>` defines the server that is running the Administration Console such as server name or IP address.

Note: The port number should be substituted with the correct port number if the default port is not used.

2. On the Administration Console Welcome page, enter your login credentials and log in.

Accepting the security certificate

When logging in, you might see a security alert with a message that says there is a problem with the security certificate. This indicates that the browser application is verifying the security certificate of the application server.

Self-signed or CA-signed certificate

The application server uses a self-signed security certificate. You might see a Security Alert when you first connect to the portal that alerts you to a problem with the security certificate. You might be warned of a possible invalid certificate and be recommended to not log in.

Although this warning appears, the certificate is valid and you can accept it. Or, if you prefer, you can install your own CA-signed certificate. For information about creating your own CA-signed certificate, see Creating a certificate authority request.

For more information about certificates, go to the IBM WebSphere® Application Server Community Edition Documentation Project at http://publib.boulder.ibm.com/wasce/V2.1.1/en/overview.html, and search for *Managing trust* and *Managing SSL certificates*.

Configuring the JDBC Driver

JDBC is an application program interface (API) specification for connecting programs written in Java™ to the data in a wide range of databases. To enable SmartCloud Cost Management Processing Engine to access a DB2® database, the appropriate JDBC drivers must be available on the server running SmartCloud Cost Management.

Note: As part of the IBM Cloud Orchestrator release, the JDBC drivers for DB2 are available and automatically configured by the installation.

Adding a JDBC driver

The appropriate JDBC driver must be available for the SmartCloud Cost Management database and any databases from which data is collected by SmartCloud Cost Management Data Collectors. JDBC is an application program interface (API) specification for connecting programs written in Java to the data in a wide range of databases. The appropriate JDBC drivers must be available on the server running SmartCloud Cost Management.

About this task

The database used to store SmartCloud Cost Management data must use the following driver:

- For DB2 for Linux db2jcc.jar and db2jcc_license_cu.jar (license JAR file), where the version is appropriate for the database to be used in the data source for SmartCloud Cost Management.

You can use other drivers for databases used by SmartCloud Cost Management Data Collectors.

Procedure

To add a new JDBC driver, copy the jar file to <SCCM_install_dir>/wlp/usr/servers/sccm/dbLibs.

Note: Restarting the server is not required.

SmartCloud Cost Management Command Line Interface

The SmartCloud Cost Management Command Line Interface (CLI), <SCCM_install_dir>/bin/sccmCLI.sh, is a generic tool that is used for querying and updating various aspects of SmartCloud Cost Management configuration.

The CLI currently supports management of data sources and data loads, which is also referred to as load tracking. The tool is Jython based and is usable both interactively and from scripts for automation. For specific details of the operations that are supported for individual areas, see the relevant sub topic for that area.

Before using the CLI, you must ensure that access credentials are provided for the Administration Console. There are two methods to do this:

- Set the relevant environment variables before running sccmCLI.sh:

  ```
  export SCCM_USER=smadmin
  export SCCM_PASSWORD=password
  ```

- When running the sccmCLI.sh script, pass the credentials on the command line:

  ```
  ./sccmCLI.sh --username smadmin --password password
  ```

Note: If neither of these options are specified, the CLI defaults to trying to connect with a user of "smadmin" and a password of "password".

The following examples show general usage for both interactive and script usage:

Interactive usage

```
./sccmCLI.sh
Python 2.5.3 (2.5:c56500f08d34+, Aug 13 2012, 14:54:35)
[IBM J9 VM (IBM Corporation)] on java1.7.0
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> sccm.dataSources()['mydatasource'].update({'host': 'myhost'})
True
>>> exit(0)
```

Usage from a shell script

```
./sccmCLI.sh <<EOF
if not sccm.dataSources()['mydatasource'].update({'host': 'myhost'}):
 exit(1)

EOF
if [ $? -eq 0 ]; then
 echo "Failed to update data source"
fi
```

Usage from a Jython script

To use the CLI from a Jython script, you must import the module required for any other standard module and add a line at the top of your script:

```
import sccm
```

If your script is not in the <SCCM_install_dir>/bin directory, ensure your **PYTHONPATH** environment variable contains <SCCM_install_dir>/bin before starting your script so the import can find the module.

Using the Command Line Interface to manage data sources

The Command Line Interface (CLI) can be used to manage data sources in SmartCloud Cost Management.

Where feasible, calls return **True** for success or **False** for failure. In the case of a failure, the error that is returned by the API call is displayed. The **sccm.dataSources()** object supports the following methods:

- **refresh()** - request the SmartCloud Cost Management server to reload the registry.tuam.xml file.
- **createDatabaseDataSource({identifier:value, property:value, ...})** - create a new Database data source.
- **createWebServiceDataSource({identifier:value, property:value, ...})** - create a new Web Service data source.
- **createMessageBrokerDataSource({identifier:value, property:value, ...})** - create a new Message Broker data source.
- **createServerDataSource({identifier:value, property:value, ...})** - create a new Server data source.
- **keys()** - return the identifiers of all available data sources

Individual data sources can be accessed by accessing **dataSources()** as a hash. Individual data sources must support the following methods:

- **test()** - test the data source.
- **delete()** - delete the data source.
- **update({property:value, ...})** - update the given properties of the data source.
- **initialise()** - initialize the data source. This is only applicable for the Database data source that is used by the Administration Console UIs.
- **upgrade()** - upgrade the datasource. This is only applicable for the Database data source that is used by the Administration Console UIs.

Additionally, the following references are available:

- **sccm.databaseTypes** - an array of database types available when creating a database data source.
- **sccm.webServiceTypes** - a hash of web service types available when creating a Web Service data source.

The following example shows a sample run:

```
$ cd etc/install/bin
./sccmCLI.sh
Python 2.5.3 (2.5:c56500f08d34+, Aug 13 2012, 14:54:35)
[IBM J9 VM (IBM Corporation)] on java1.6.0
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> ds = sccm.dataSources()
>>> print ds
{'os_keystone': WebService, 'os_qpid_default': MessageBroker, 'SCCM': Database}
>>> print ds['SCCM']
{'about': 'https://localhost:9443/sccmOslc/dataSources/databases/SCCM',
 'actionTestDataSource': 'https://localhost:9443/sccmOslc/dataSources/databases/SCCM?oslc_ua.actio
 'database': None,
 'databaseType': '4',
```

```
 'host': 'localhost6',
 'identifier': 'SCCM',
 'isAdministrationDataSource': True,
 'isDriverLoaded': True,
 'isProcessingDataSource': True,
 'isTestable': True,
 'objectPrefix': 'DB2INST1.',
 'port': '1234',
 'type': 'Database',
 'userName': 'db2inst1'}
>>> ds['SCCM'].test()
400: Error getting connection
AUCCM5022E An error was detected in the data layer. The following information was provided:
 Connection refused. Review the trace log to get detailed information.
False
>>> ds['SCCM'].update({'port':60004})
True
>>> ds['SCCM'].test()
True
>>> ds.refresh() # Ask server to reload registry.tuam.xml if manually modified
>>> ds['SCCM'].delete()

>>> print sccm.databaseTypes
['Microsoft SQLServer', 'Oracle on Windows', 'Oracle on Linux/Unix', 'DB2 on Windows', 'DB2 on Linux,
>>> print sccm.webServiceTypes
{'Other': 'OTHER', 'VMware': 'VMWARE', 'REST': 'REST'}

>>> ds.createDatabaseDataSource({'identifier':'dsname', 'databaseType':'Oracle on Windows', 'title':
```

Using the Command Line Interface to manage load tracking

The Command Line Interface (CLI) can be used to manage load tracking in SmartCloud Cost Management.

Where feasible, calls return **True** for success or **False** for failure. In the case of a failure, the error that is returned by the API call is displayed. The **sccm.dataLoads()** has a single attribute that is called **feeds**, which contains all the available feeds, addressable as a hash map. Each individual field has two attributes that are called **years** and **loads**. **Years** can be filtered at various levels of granularity as follows:

- **dl.feeds** - Available feeds, addressable as a hashmap.
- **dl.feeds['SCO']** - Detail for the **SCO** feed.
- **dl.feeds['SCO'].years[2012]** - Filtered for the year 2012.
- **dl.feeds['SCO'].years[2012][12]** - Filtered for accounting period 12 in year 2012.
- **dl.feeds['SCO'].years[2012][12]['2011-12-28T00:00:00Z']** - Filtered for a specific date in accounting period 12 in year 2012.

At each filter level, the **loads** attribute provides all the loads that match that filter. A **loads** object can be further filtered by load identifier to get more information about that load. The loads object also supports a **delete** method, which takes a single true or false flag to indicate whether a full delete must be performed. **False** deletes the database content only. **True** deletes both the database content and the load tracking entry.

The following example shows a sample run:

```
$ cd <SCCM_install_dir>/bin
./sccmCLI.sh
Python 2.5.3 (2.5:c56500f08d34+, Aug 13 2012, 14:54:35)
[IBM J9 VM (IBM Corporation)] on java1.6.0
```

```
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> dl = sccm.dataLoads()
>>> print dl.feeds.keys()
['TSM', 'LINUXKVM', 'NODBSIZE', 'EvtPrt', 'VMWARE', 'TPC', 'MSSQL2K', 'TUAMHMC', 'TPCPOOL', 'SCO']
>>> print len(dl.feeds['SCO'].loads)1727
>>> print len(dl.feeds['SCO'].years[2012].loads)
1098
>>> print len(dl.feeds['SCO'].years[2012][11].loads)
90
>>> print dl.feeds['SCO'].years[2012][11].loads
{1880: Summary,
 1881: Detail,
 1882: Ident,
 1883: Summary,
...
 1967: Summary,
 1968: Detail,
 1969: Ident}
>>> print dl.feeds['SCO'].years[2012][11].loads[1968]
{'accountCodes': ['Project Q                george.green.us@email.comVM740607926
                  'Project N                liam.green.us@example.comVM727427171
                  'Project U                ringo.black.ie@example.coVM117361024
...
                  'Project Z                john.white.ie@example.comVM185524792
                  'Project Z                john.green.de@email.com  VM189152095
                  'Project Z                john.jones.us@email.com  VM508743332
 'groupIdentifier': '1098',
 'identifier': 1968,
 'totalRecords': '282',
 'type': 'Detail  '}
>>> dl.feeds['SCO'].years[2012][11].loads[1968].delete(False)
True
>>> print dl.feeds['SCO'].years[2012][11].loads[1968]
/sccmOslc/dataLoads/feeds/SCO/loads/1968
{'accountCodes': None,
 'groupIdentifier': '1098',
 'identifier': 1968,
 'totalRecords': '282',
 'type': 'Detail  '}
>>> dl.feeds['SCO'].years[2012][11].loads[1968].delete(True)
True
>>> print dl.feeds['SCO'].years[2012][11].loads[1968]
None
>>> dl.feeds['SCO'].years[2012][11].loads.delete(True)
True
```

Configuring the SmartCloud Cost Management data sources

A data source is required to connect to the SmartCloud Cost Management database. A data source is also required for SmartCloud Cost Management Data Collectors that collect data from a database or a Web service.

Important: The SmartCloud Cost Management data sources are automatically configured by running the `ico_configure.sh` script after SmartCloud Cost Management is installed. For more information about this script, see the related topic. However, if required you can manually create the data sources as explained in this section.

There are four types of data sources in SmartCloud Cost Management:

- **Database**: For DB2 databases that use the supported default drivers, these are the data sources that connect to a database that you are collecting data from using a SmartCloud Cost Management Data Collector.

- **Message broker**: These are the data sources that connect to a Message Broker that you are collecting data from using a SmartCloud Cost Management Data Collector.
- **Server**: These are the data sources that connect to a server that you are collecting data from using a SmartCloud Cost Management Data Collector.
- **Web Service**: These are the data sources that connect to a Web service that you are collecting data from using a SmartCloud Cost Management Data Collector.

Note: An **All** option is also available in the **Data Source Type** menu. Select this option if you want to view all data source types. When the **All** option is selected, the **Create Data Source** button is disabled.

Creating data sources

Note: Data source information is stored in `<SCCM_install_dir>/config/registry.tuam.xml`. Manual editing of this file is not recommended. Triple DES and SHA-1 are used to secure credential information.

Adding a Database data source

A Database data source is used to connect to a SmartCloud Cost Management database. A Database data source is used to connect to a DB2 for Linux, UNIX, and Windows; DB2 for z/OS®; Oracle; or SQL Server database that you are collecting data from using a SmartCloud Cost Management Data Collector.

Before you begin

Use the **Database** data source type to create data sources for DB2 for Linux, UNIX, and Windows; DB2 for z/OS; Oracle; or SQL Server databases that use the supported default drivers that are described in "Configuring the SmartCloud Cost Management data sources" on page 13.

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Database** as the **Data source Type**.
2. Click **Create Data Source**.
3. Complete the following:

   Note: All fields marked with an * are mandatory and must be completed.

   Data Source Name
   Type the name that you want to assign to the data source.

   Note: The following are invalid characters for a data source name: "/", "\","","":","?","<",">",".","|",".".

   Username
   Type the database user ID.

   Password
   Type the database password.

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

   If you are using an Internet Protocol Version 6 (IPv6) address as the host name, the IP address must be specified as follows:

- Enclose the address with square brackets. For example, IPv6 address `aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb` should be specified as `[aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb]`.

Database Name
Type the name of the database that you want the data source to point to.

For a DB2 for z/OS data source, the field contains a two-part entry of <location name>/<database name>. To determine the correct location name, refer to the DDF configuration that is displayed in the z/OS startup messages as shown in the following example:

```
13.17.59 STC16980  DSNL003I  :D81L DDF IS STARTING
13.18.21 STC16980  DSNL004I  :D81L DDF START COMPLETE  611
   611                        LOCATION  KSCDB201
   611                        LU        USCACO01.DB2D81L
   611                        GENERICLU -NONE
   611                        DOMAIN    demomvs.db2.ibm.com
   611                        TCPPORT   446
   611                        RESPORT   5020
```

In this example, the location is `KSCDB201`. If you create a DB2 for z/OS database named `TUAM71`, you would type `KSCDB201/TUAM71` in this field.

Database Type
Select the type of database.

Object Prefix
For all database types other than Microsoft SQL Server, type the schema name for the database. This value is case-sensitive. Therefore, if the schema name is SmartCloud Cost Management, type `SCCM` and not `sccm`. Database schemas are defined using database administration tools. If you do not know the schema name, consult your database administrator.

For SQL Server, it is recommended that you type `dbo.`. This object prefix sets the owner of the database objects in the database to dbo, which allows any authorized database user to view the objects.

Driver Loaded
If checked, this field Indicates that the database JDBC driver is loaded on the classpath and when the database data source is configured, it will connect to the database. If unchecked, then the JDBC driver must be configured. See the related topic for more information.

Port By default, SmartCloud Cost Management will connect to one of the following ports on the database server: 50000 (DB2 for Linux, UNIX, and Windows); 446 (DB2 for z/OS); 1433 (SQL Server); or 1521 (Oracle). If you are using a port other than one of these default ports, type the port number.

Parameters
Type any additional parameters that are required to enable connection to the database.

Database URL
Type a URL if you want to use a URL other than the default. For example, you want to add properties to the URL.

4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

Note: When the data source information is saved, the connection to the database is verified. You should see a message at the top of the screen indicating that the connection was successful.

Setting the Default Admin and Processing data source

If you are using multiple databases to store SmartCloud Cost Management data (for example, you have a production database and a development database) you must select the data source for one database as the default for administration and data processing. Note that this process is applicable for Database data sources only.

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Database** as the **Data Source Type**.
2. Select the required data source name from the **Data Source Name** menu.
3. Click each of the following:

Default Admin
Indicates whether the data source is currently used by the SmartCloud Cost Management Administration Console application. If the **Default Admin** checkbox is checked, this is the data source that the Administration Console will use.

Default Processing
Indicates whether the data source is currently used by the SmartCloud Cost Management Job Runner utility. If the **Default Processing** checkbox is checked, this is the data source that Job Runner will use.

Adding a Web service data source

A Web Service data source is used to connect to a Web service that you are collecting data from using a SmartCloud Cost Management Data Collector. This topic provides the steps to create Web Service data sources.

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Web service** as the **Data Source Type**.
2. Click **Create Data Source**.
3. Complete the following:

Note: All fields marked with an * are mandatory and must be completed.

Data Source Name
Type the name that you want to assign to the data source.

Note: The following are invalid characters for a data source name: "/", "\","","":","?","<",">",".","|",".".

Username
Type the Web service user ID.

Password
Type the Web service password.

URL Type the Web service URL as follows, using either the http or https protocol as required:.

`http://<Server Name>:port`

Or

`https://<Server Name>:port`

Web Service Type
Select **Other**, **VMware**, or **REST** as the web service type.

Keystore File
The Keystore file contains the vCenter or REST server certificate that is used for authentication during the secure connection between the collector and the vCenter web service. The password is used to access the file. Enter a valid path to the file.

Keystore Password
Type the Keystore password.

4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the vCenter or REST server is verified. You should see a message at the top of the screen indicating that the connection was successful.

Adding a Server data source

A Server data source is used to connect to a server that you are collecting data from using a SmartCloud Cost Management Data Collector. This topic provides the steps to create Server data sources.

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Server** as the **Data Source Type**.
2. Click **Create Data Source**.
3. Complete the following:

   Note: All fields marked with an * are mandatory and must be completed.

   Data Source Name
   Type the name that you want to assign to the data source.

   Note: The following are invalid characters for a data source name: "/", "\","","":","?","<",">",".","|",".".

   Username
   Type the server user ID.

   Password
   Type the server password.

   Host Type the host name, IP address, or IP name where the database resides. If you are using an Internet Protocol Version 6 (IPv6) address as the hostname, the IP address must be specified as follows:

   - Enclose the address with square brackets. For example, IPv6 address `aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb` should be specified as `[aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb]`.

   Protocol
   Enter the connection protocol, SSH or TCP. If this field is left empty, it defaults to SSH.

   Port Enter the port number. If this field is left empty, it defaults to 22. This number can be updated if you are using the non default Port number.

Timeout
Enter the timeout value. If this field is left empty, it defaults to 180000. This number can be updated if you want to use a non default timeout value.

Private Key File
The private key file is a file which contains an encrypted key for authenticating during a secure connection. The passphrase is used to encrypt or decrypt the key file. Enter a valid path to the key file.

Restricted Shell
Select this check box if the connection to the server is a restricted shell type.

4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the database is verified. You should see a message at the top of the screen indicating that the connection was successful.

Adding a Message Broker data source

A Message Broker data source is used to connect to a Message Broker that you are collecting data from using a SmartCloud Cost Management Data Collector. This topic provides the steps to create Message Broker data sources.

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Message broker** as the **Data Source Type**.
2. Click **Create Data Source**.
3. Complete the following:

   Note: All fields marked with an * are mandatory and must be completed.

   Username
   Type the message broker user ID.

   Password
   Type the message broker password.

   Host Type the host name or IP address where the message broker resides.

   Broker Type
   Select the type of message broker that you want to use. The default value is `Qpid` .

   Client ID
   Type the client id for the message broker. Client id is an identifier property stipulated by the JMS API specification which is supported by AMQP message brokers.

   Virtual Host
   Type the Virtual host for the message broker. The Virtual host is a path that acts as a namespace that is used to partition the message broker data into distinct sets.

   Protocol
   Select the transport protocol that the message broker must use. The default value is `TCP`.

Port Type the Port number used by the message broker. The default value is 5672.

Timeout
Type the timeout value of how long (in milliseconds) to wait for the connection to succeed. The default value is 180000.

SSL Enabled
Select this check box if the connection to the message broker is over Secure Sockets Layer (SSL).

4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the data source is verified. You should see a message at the top of the screen indicating that the connection was successful.

Maintaining data sources

To list data sources and optionally delete old data sources that do not work anymore, use the `datasourceManager.py` utility.

Run the following commands:

```
cd <SCCM_install_dir>/bin
./datasourceManager.py [--delete-broken --yes]
```

If you run the utility with no arguments, it lists all data sources and it shows whether a connection could be established. If you specify the `--delete-broken` and `--yes` arguments, all the broken data sources are deleted.

About initializing the database

Initializing the database prepares the database for use by SmartCloud Cost Management. Initializing will overwrite any existing data. You will receive a confirmation message if you attempt to initialize a database that has already been initialized.

Important: The database is initialized automatically by running the ico_configure.sh script after SmartCloud Cost Management is installed. For more information about this script, see the related topic. However, if required you can follow the steps in this section to manually initialize the database.

Initializing the database performs the following tasks:

- Creates new database tables
- Populates these tables with an initial set of data
- Creates necessary database objects

Initializing the database

Initializing the database prepares the database for use by SmartCloud Cost Management. Initializing will overwrite any existing data. You will receive a confirmation message if you attempt to initialize a database that has already been initialized.

Before you begin

If you are using multiple databases for SmartCloud Cost Management, make sure that the data source for the database that you want to initialize is set to **Default Admin**. Initializing overwrites any existing data in a database connected to the data source.

Note: If the user ID that you are using to access the database does not have sufficient system administration authority, SmartCloud Cost Management might not be able to create database objects or to create objects with the appropriate permissions during database initialization. In this situation, a warning message is displayed recommending that you contact your database administrator before continuing.

About this task

Procedure

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Database** as the **Data Source Type**.
2. Use the **Data Source Name** menu to select the required database name and click **Initialize Database**.

Loading the database with sample IBM Cloud Orchestrator data

You can load the database with sample IBM Cloud Orchestrator data using the `RunICOSamples.sh` script in the `<SCCM_install_dir>/bin` directory. This script runs the sample job file `SampleOpenStackDemoData.xml` provided with SmartCloud Cost Management in the `<SCCM_install_dir>/samples/jobfiles` directory.

You are not required to load sample data. The `RunICOSamples` script is used only to verify the installation and create sample data that can be viewed in Administration Console and in reports. Once the SmartCloud Cost Management is being used with live production data, the sample data should be deleted using the Load Tracking page in Administration Console.

Before running the script it is advised to import all the required rates using the Import Rate Templates feature and assign each rate a rate value. The templates to import are:

- Virtual Systems
- License
- Charges
- Hosting Charges with VM Sizes
- Virtual Images

The `RunICOSamples` script calls the job file using two `-date` parameters. The `-date` parameters determines the start and end dates of the output data. For example, if the first `-date` parameter is set to yyyy0601 and the second `-date` parameter is set to yyyy1231 (where yyyy is the year), the start date for the data is `June 01, yyyy`

and the end date is December 31, yyyy. If no parameters are included, default parameters defined in the script are used. These are the current dates and the current date minus 60 days. For example, if todays date is Sept 12 2013, the start date is set to 20130714 and the end date is set to 20130912 and the data is loaded for these dates.

The following are the commands for running the RunICOSamples script in a Linux environment:

- **Linux:** Using a shell command, type: <SCCM_install_dir>/bin/RunICOSamples.sh

Security overview

This topic describes the security features for SmartCloud Cost Management.

Security in SmartCloud Cost Management is governed by the following mechanisms:

- Users and roles defined in IBM Cloud Orchestrator determine access to the Administration Console and reports in Jazz for Service Management.
- Users, User Groups and Clients defined in the SmartCloud Cost Management application control Account Code security.
- User Roles defined in Jazz for Service Management.
- Cognos based Tivoli Common Reporting security.

Account Code security for IBM Cloud Orchestrator

Account code security in SmartCloud Cost Management is used for domain and project reporting segregation based on the cloud roles in IBM Cloud Orchestrator.

Account code security restricts the report data that a user can view by associating SmartCloud Cost Management clients and users to user groups.

Account code security allows you to access a controlled set of SmartCloud Cost Management reporting data. Access to usage data is controlled by the association of a user with accessible client account codes through the use of a user group. As a result, the client account codes that a user can view in reports is restricted based on the clients that are assigned to the user group. In SmartCloud Cost Management, a user can belong to one or more user groups.

SmartCloud Cost Management provides reporting access to the members of two IBM Cloud Orchestrator security roles:

- Cloud Administrators (admin)
- Domain Administrators (domain_admin)
- Project Members (Member)

For more information on security roles in IBM Cloud Orchestrator, see the related topic.

In IBM Cloud Orchestrator, these two security groups define what a users functional role is. The following table describes what access users with these roles have to SmartCloud Cost Management report data. This access is granted after SmartCloud Cost Management clients, users, and user groups have been generated and associated together for them by the OpenStackKeystoneContext job in the OpenStackContext job file. For more information about how this job works see the related topic.

Table 1. Access for users

IBM Cloud Orchestrator security groups	SmartCloud Cost Management report data access restrictions
Cloud Administrators (admin)	Can access IBM Cloud Orchestrator reporting data in SmartCloud Cost Management for all domains and projects.
Domain Administrators (domain_admin)	Can access IBM Cloud Orchestrator reporting data in SmartCloud Cost Management for all projects belonging to their domain.
Project Members (Member)	Can access IBM Cloud Orchestrator reporting data in SmartCloud Cost Management for all projects that they are members of.

The `CreateAccountRelationship` and `CreateUserRelationship` stages are used to automate Account code security in reporting for the IBM Cloud Orchestrator roles mentioned above. This is done by generating SmartCloud Cost Management clients, users and user groups and associating them together. For more information about these stages, see the related topics.

REST APIs can be used to work with account code structures, clients, user groups and users. For more information about REST APIs, see the related topic.

IBM Cloud Orchestrator Account code structure

The IBM Cloud Orchestrator Standard account code structure is set as the default for all SmartCloud Cost Management cloud user groups. It is also the base format for creating all the SmartCloud Cost Management clients created by the `CreateAccountRelationship` stage of the `OpenStackContextKeystone` job.

An account code structure reflects the chargeback hierarchy for the organization. SmartCloud Cost Management uses an account code to identify entities for billing and reporting. This account code determines how SmartCloud Cost Management interprets and reports input data. An **`Account_Code`** identifier is added to the various IBM Cloud Orchestrator CSR feeds in a format that reflects the IBM Cloud Orchestrator Standard account code structure. The IBM Cloud Orchestrator Standard account code structure is as follows:

- `<DOMAIN>` - 25 characters
- `<PROJECT>` - 25 characters
- `<USER>` - 25 characters
- `<RESOURCE>` - 32 characters

This account code format is generated and processed by the sample IBM Cloud Orchestrator job files, `OpenStackImages.xml`, `OpenStackVMInstances.xml`, and `OpenStackVolumes.xml`. This structure defines the account code levels that appear in invoices and other reports and can be drilled through to get a chargeback break down from any accounting level perspective in the predefined structure.

User management

This section provides information required to configure a Central User Registry in order to ensure that an organizations or systems existing users can be granted access to the portals used by SmartCloud Cost Management

Configuring Keystone as a Central User Registry

To ensure that the users created in IBM Cloud Orchestrator are available in both the SmartCloud Cost Management Administration Console and in Jazz for Service Management, you can configure Keystone as a Central User Registry on both of these systems.

Important: The configuration of Keystone as a Central User Registry is automatically done by running the `ico_configure.sh` script after SmartCloud Cost Management is installed. For more information about this script, see the related topic. However, if required you can manually configure Keystone as a Central User Registry as explained in this section.

After installation, only Cloud Administrators can log into the SmartCloud Cost Management Administration Console to administer the system.

Configuring Keystone as a Central User Registry on both SmartCloud Cost Management Administration Console and Jazz for Service Management is done as follows:

- Keystone is configured automatically on the SmartCloud Cost Management Administration Console as part of installing SmartCloud Cost Management.
- Keystone is configured automatically on Jazz for Service Management as part of the ico_configure.sh post installation configuration script. For more information about this script, see the related automated configuration topic.

However, if required, you can also configure Keystone manually in Jazz for Service Management using the following steps:

1. Log on to the SmartCloud Cost Management system and run the following configuration command:

   ```
   <SCCM_install_dir>/bin/postconfig/enableKeystoneUserRegistry.sh --host hostname --user smadmin
   ```

   where:

   - `hostname` – the host name of the Jazz for Service Management server.
   - `smadmin` – the Jazz for Service Management administration user.
   - `smpass` – the Jazz for Service Management administration user password.

Note: If required, you can override the default Keystone shared secret using the `--keystone-password` option.

After running this configuration command, you can define Jazz for Service Management permissions for users. For more information about this, see the related topics.

After the Keystone is configured as a Central User Registry on both systems, the credentials for users that are created in IBM Cloud Orchestrator by using Keystone can be used when logging in to both of these systems. See the related topic for information about managing users in IBM Cloud Orchestrator.

Defining Tivoli Common Reporting security permissions

Use the following topics to access information about security settings in Tivoli Common Reporting.

Note: Report Level Security for Tivoli Common Reporting is managed within Cognos itself. Data Level Security is managed within SmartCloud Cost Management.

Managing Jazz for Service Management roles for users

This topic provides the steps required to add Jazz for Service Management roles to users. Once Keystone is configured as a Central User Registry for Jazz for Service Management, all IBM Cloud Orchestrator users will be able to log into the portal.

About this task

As part of the automated post configuration script ico_configure.sh, (see related topic), all IBM Cloud Orchestrator and Jazz for Service Management users are granted the `tcrPortalOperator` role. The script configures Tivoli Common Reporting so that only the default admin user, for example, `smadmin`, can administer Tivoli Common Reporting and all other users can access and run reports.

Additional privileges are required to allow users to create reports. These privileges must be assigned to the relevant users within Tivoli Common Reporting.

Note: Although Jazz for Service Management users can run reports, the data in those reports is restricted by account code security. For more information about account code security, see the related topic.

Only the default admin user can create and amend custom reports by default, so if users want to create them, they must be assigned the `Authors` role in Cognos. For information about how to do this, see the related configuring security permission topic.

The automated configuration script (see related topic) only grants the `tcrPortalOperator` role to the users that exist at the time it is run. New users created in the Central User Registry do not have access to Tivoli Common Reporting unless access is assigned to them within Jazz for Service Management. A script called `configure_tcr_users.sh` is available to assign access to Tivoli Common Reporting to all users. This script can be scheduled to run nightly using a `CRON` job to ensure all users are given access. This is an optional configuration step.

The `configure_tcr_users.sh` script is located in the following directory in the Jazz for Service Management installation: `<JazzSM_home_dir>/reporting/sccm/bin`. The script is run by specifying the following options:

```
--jazz    <Jazz for Service Management Host>
--jazzuser   <Jazz for Service Management User>
--jazzpass   <Jazz for Service Management Users Password>
```

The following shows an example of how this script can be run:

```
./configure_tcr_users.sh --jazzuser smadmin --jazzpass mypassword --jazz myhost
```

Alternatively, if you want to manually add a role to a user in Jazz for Service Management, complete the following steps

Procedure

1. Open the **User Roles** panel:
   - In Jazz for Service Management – click  **> User Roles**.
2. On the **User Roles** page, click **Search** to find all users or type a user ID and click **Search** to find a specific user in the Central User Registry.
3. Click on the user you want to assign a role to.
4. Check the role or roles you want to assign to the user.

   The roles that are applicable to administer Tivoli Common Reporting and the portal itself are:
   - tcrPortalOperator
   - iscadmins
5. Click **Save**. This saves directly to the master configuration.

Configuring security permissions

Increase the security settings for the Cognos based Tivoli Common Reporting user permissions using the Administration Console. After the ico_configure.sh post install configuration script is run, all the users except for the default admin user will have the `Consumer` role in Cognos. This role allows users to run and schedule reports. The default admin user has the `System Administrator` role, which allows them to administer the system, including assigning access, and developing custom reports. See the related topics for more information about the ico_configure.sh script and defining portal permissions for users.

About this task

For information about Tivoli Common Reporting version 3.1.0.1 security settings for authorizations, see Business Intelligence Administration and Security Guide 10.2.0.

By default, all new users created for the Common reporting portlet are assigned to `Consumers` user group which allows them to run and schedule reports. To increase the security of your reporting solution, edit the members of the security settings in the Cognos Administration Console. Cognos comes with a set of predefined roles which can be used to assign users access to various capabilities or alternatively, access to these capabilities can be assigned directly. The following procedure describes how to assign users to a role, such as the `Authors` role, which is used to access to Report Studio for custom reports:

Procedure

1. Log in to the reporting interface. Based on the version of Tivoli Common Reporting you are running, refer to the related topic.
2. In the **Common Reporting** window, click **Administration** from the **Launch** drop-down list.
3. Click the **Security** tab, go to **Users, Groups, and Roles**, and select the Cognos user namespace.
4. Locate the `Authors` group, and set properties for the group by clicking **More** > **Set properties**.
5. On **Members** tab, click **Add** to add an individual administrative user.
6. Add the administrative user of your choice from the `VMMProvider` namespace, and click **OK** to save the settings.
7. Click **OK** to save the new settings.

8. To assign users access to capabilities directly, such as Report Studio, do the following:
   a. Log in to the reporting interface. Based on the version of Tivoli Common Reporting you are running, refer to the related topic.
   b. In the **Common Reporting** window, click **Administration** from the **Launch** drop-down list.
   c. Click the **Security** tab, go to **Capabilities**.
   d. Locate the Capability that must be granted access to and click on the down arrow to set properties for the capability.
   e. On the **Permission** tab, click **Add** to add an individual user.
   f. Add the administrative user of your choice from the `VMMProvider` namespace, and click **`OK`** to save the settings.
   g. Assign the relevant grants to the user.
   h. Click **OK** to save the new settings.

Constraining access to reports

Manage permissions granted to users or user groups for reports, and capabilities for reports, report sets or folders in the same way as using native Cognos concepts and methods. By default, permissions and capabilities that user groups or reports are assigned to are inherited from the parent entry.

About this task

Note: The users and groups referred to here are Central User Registry users and groups and not SmartCloud Cost Management users and groups.

You can change the default permissions that specific groups or users have to reports or report packages. You can also change capabilities for reports, report sets and folders.

Procedure

1. Log in to the reporting interface. Based on the version of Tivoli Common Reporting you are running, refer to the related topic.
2. In the **Common Reporting** window, navigate to the report for which you want to change user permissions and select it.
3. Click **Actions** > **Set properties**.
4. Go to the **Permissions** tab. The table shows default permissions set for user groups.
5. Select **Override the access permissions acquired from the parent entry** and choose the types of permissions that you want to grant to specific user groups.
6. Go to the Capabilities tab. In the table you can see what capabilities are assigned to reports, report sets or folders.
7. Select **Override the capabilities acquired from the parent entry** to grant and deny capabilities.

What to do next

To find out more about permissions and capabilities, see Cognos Connection User Guide 10.2 Guide .

Chapter 3. Administering the system

To administer and manage SmartCloud Cost Management, you must set up rate codes and rate groups, set up the calendar, and set configuration options.

Defining offerings and rate groups

Using this section define offerings and rate groups.

Offerings

Offerings make it easier to manage rate groups and rates by logically categorizing related rate groups and rates within a rate table. Rate groups and its contained rates can be optionally categorized into offerings. Rate templates that are imported into rate tables are automatically categorized into offerings that reflect the rate template.

Rate Groups

All rate codes must be assigned to a group. Creating and using rate groups lets you create rate subtotals in reports, graphs, and spreadsheets. Grouping rates such as Mainframe charges, Windows charges, and UNIX charges allows reports to be summarized in a way that is meaningful.

Rate groups allow users to create reports that are based on grouping rates that have the same identifier or identifiers. It is advised that you do not assign rate codes with different identifiers to the same rate group. Combining rates with different identifiers within the same group results in reporting anomalies. It is advised that separate rate groups are created for each resource file type. For example, create a UNIX charges group for UNIX resource files with the same identifiers. This ensures that the rates within the group have the same identifiers.

Adding offerings

All rate groups must be assigned to an offering. If a rate group is not assigned to an offering, it is displayed under the **Unassigned** offering node in the tree-table.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. To enable the **Create Offering** button, you must select the top-level node in the tree table first, as the button is not enabled by default.
3. Click **Create Offering**. Alternatively, you can right-click on the top-level node in the tree table and click **Create Offering**.
4. In the Create Offering dialog box, complete the following fields:

   Name Enter the name for the offering.
5. Click **Apply** to save the offering name, or **Cancel** to return to the Rate Groups page.
6. If you clicked **Apply**, a message is displayed indicating that the offering is created. Click **OK** to return to the Rate Groups page, where the new offering is displayed.

© Copyright IBM Corp. 2013, 2015

Deleting offerings

Use the Rate Group Maintenance page to delete offerings that are no longer required.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. To enable the **Delete Offering** button, you must select the offering row in the tree table first, as the button is not enabled by default.
3. Click **Delete Offering** to delete the selected offering. Alternatively, you can right-click on the offering row and click **Delete Offering**.
4. Click **Yes** to delete the offering or **No** to return to the Rate Groups page.

   Note: When an offering is deleted, all its rate groups and associated rates are moved to the **Unassigned** offering node in the tree table.

Adding rate groups

All rate codes must be assigned to a group. Grouping rates such as Mainframe charges, Windows charges, and UNIX charges allows reports to be summarized in a way that is meaningful.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. To enable the **Create Rate Group** button, you must select the offering row in the tree table first, as the button is not enabled by default.
3. Click **Create Rate Group**. Alternatively, you can right-click on the offering row and click **Create Rate Group**.
4. In the Create Rate Group dialog box, complete the following fields:

   Name Type the name that you want to assign to the rate group.

   Description
   Type a description for the rate group. This is the value that is shown on the Rate Groups page and the value that is displayed for the rate group in the standard reports that are provided with SmartCloud Cost Management. If the description of rate group is different from its associated rate code, then both descriptions are displayed in the tree with the rate group description enclosed in brackets.
5. Click **Apply** to save the rate group name and description or click **Cancel** to return to the Rate Groups page.
6. If you clicked **Apply**, a message is displayed indicating that the rate group has been created successfully. Click **OK** to return to the Rate Groups page, where the new rate group is displayed.

Changing the rate group sequence

The sequence that the rate groups are displayed in the tree-table is the sequence that the groups are displayed in the reports. You can sequence the rate groups in any order.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. Expand the required offering and locate the rate group whose sequence you want to change.
3. Click the rate group and drag it to the new location in the rate group sequence within its associated offering.

   Note: When the drop box is green, it means that you can move the rate group to that location, for example, either before or after a rate group. If the drop box is red, it means that you are not allowed to move the rate group to that location, for example, if you drag a rate group into another rate group.
4. When the rate group is moved to another location, either in the same offering or another offering, its new sequence location is saved automatically.

Deleting rate groups

Use the Rate Group Maintenance page to delete rate groups that are no longer required.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. To enable the **Delete Rate Group** button, you must select the rate group row in the tree table first, as the button is not enabled by default.
3. Click **Delete Rate Group** to delete the selected rate group. Alternatively, you can right-click on the rate group row and click **Delete Rate Group**.
4. Click **Yes** to delete the rate group or **No** to return to the Rate Group Maintenance page.

   Note: When a rate group is deleted, all its rates are moved to the **All Unassigned** folder.

Defining rates using the Rate Tables panel

In SmartCloud Cost Management 2.1.0.5 rates are managed using the **Rate Tables** panel. The main function of the **Rate Tables** panel is to manage standard rates, however, it also introduces features such as historical and tiered rating. These features are fully supported by Cognos based Tivoli Common Reporting.

Adding rate tables

This topic describes how to create rate tables.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables** .
2. Click **Create Rate Table**.
3. In the **Rate Table Properties** section, complete the following:

   Rate Table Name
   The rate table name.

   Description
   Shows the description for the rate table.

   Effective From
   The effective from date for new rates created in the rate table. The rates specified in the rate table cannot have an effective date before this date. An empty value indicates that the rates can have any effective date.

   Expiry On
   The default expiry date for new rates created in the rate table. The rates specified in the rate table cannot be effective after this date. An empty value indicates that the rates can be set to not expire.

   Lockdown Date
   The **Lockdown Date** shows the date before which effective rates cannot be created, altered, or removed.

   - **Actual** - Enter the actual lock down date in this field. This date is a fixed point in time date.

   Currency Symbol
   The currency symbol that is used when creating any monetary rates in the rate table.
4. Click **Create** to create the rate table.

Related tasks:

Importing rates from an existing rate table
Use the Import Rates feature to import rates from an existing rate table into another rate table for a specific effective date. You can use the same rates that were used in an existing rate table instead of manually creating them each time and associating them with the new rate table.

Removing rate tables

This topic describes how to remove a rate table.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables** .
2. Select the rate table that you want to remove in the **Rate Table Name** drop down menu.
3. Once selected, click **Remove Rate Table**.

   CAUTION:
   In addition to removing the rate table, all past, present and future effective rates, tiers and rate shifts associated with the rate table are also removed. Once a rate table has been removed, the action cannot be reversed.
4. Click **Yes** to remove the rate table or **No** to cancel the deletion.

Defining rates

SmartCloud Cost Management uses a rate, which is represented by a rate code, to calculate a cost for each resource that is reported. Examples of resources are CPU time used, jobs started, data received or sent, disk space used and lines printed.

The CSR and CSR+ files used by SmartCloud Cost Management contain rate codes. Rate codes represent the resource units that are reported (CPU time used, jobs started, data received, or sent, disk space used, lines printed, and so on). To enable SmartCloud Cost Management to process and report the rate codes in the CSR or CSR+ file, the codes must be defined in SmartCloud Cost Management. The definition for each rate code includes a monetary value for the rate code and other rate processing information.

Many of the rates produced by SmartCloud Cost Management are pre-loaded in the `STANDARD` rate table. You can then use the date search functionality to locate rates to delete that you do not want to use or add rates that are not included in the `STANDARD` rate table or any other rate table for the requested date. You can use either the **Date** or **Date Range** options to locate rates you want to modify the values or advanced properties of. If you need to perform differential costing, for example, you want to charge Client A different rates than Client B, you can create other rate tables in addition to the `STANDARD` table using the **Create Rate Table** option.

Viewing rates

This topic describes how to view rates that are effective on a specific date or across a date range.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Tables** .
2. On the Rate Tables page, select the required table from the **Rate Table** dropdown menu.
3. Use the date search functionality to locate rates that are effective on a specific date or rates that are effective across a date range.
4. To search for rate codes associated with the selected rate table for a specific effective date, complete the following fields:

   Date Select this option when looking for rates that are active on a specific date.

   Effective Date - When the Date option is selected.
   A list that contains all of the effective dates for the different sets of rates that exist in the selected rate table. The list also contains a **New Effective Date** option used to create a new effective date for a new set of rates. When looking for rates for a specific date, select the **Date** radio button option and an existing effective date in the **Effective Date** field. Once the date is entered, a tree table is displayed with rate information for the date selected.

   The tree table is populated with the rates that are effective on the date specified.
5. To search for rates associated with the selected rate table across a particular date range, complete the following fields:

Date Range
Select this option when looking for rates for a particular date range.

Effective From - When the Date Range option is selected.
A list that contains all of the effective dates for the different sets of rates that exist in the selected rate table. When looking for rates for a date range, select the **Date Range** radio button and enter an existing effective date in the **Effective From** field. This field indicates the lowest effective date in a date range, when searching for rates to display in the main tree table. This field is used in conjunction with the **Effective To** field to create the date range search criteria. Once the **Effective From** and **Effective To** dates are entered, a tree table is displayed with rate information for the date range selected.

Effective To
A list that contains all of the expiry dates for the different sets of rates that exist in the selected rate table. It is used to indicate that rates with an effective date after this date are to be excluded from the date range search. This field is used in conjunction with the **Effective From** field to create the date range search criteria. Once the **Effective From** and **Effective To** dates are entered, a tree table is displayed with rate information for the date range selected.

The tree table is populated with the rates that are effective within the date range specified.

Results

Once the rates have been found, you can proceed to create new rates, remove or modify them.

Adding rates

This topic describes how to add normal rates to a rate group.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. On the Rate Tables page, select the required table from the **Rate Table** dropdown menu.
3. Select the **Date** option.

   Note: The **Date** option must be selected when creating or removing rates. When the **Date** option is selected the **Effective Date** field is displayed.
4. In the **Effective Date** drop down menu, select an existing effective date or click **New Effective Date** to create a new effective date for a new set of rates.
5. If the **New Effective Date** option is selected, enter the date in the **New Date** field. Once an effective date is selected, a tree table is displayed with rate information for the date selected.
6. In the **Manage Rate Structure** tree table, expand or collapse the **Id** option to drill down through the hierarchy of different rate-related object Ids. The first node in the tree is the offering, then the rate group, then the rate followed by either rate tiers or rate shifts if they exist.
7. Right click on the required rate group and select **New Rate**. An action dialog panel is displayed with the following fields:

Note: In this task, you want to create normal rates. When creating normal rates, the rate patterns described in this section apply.

- **Name** - The rate name which represents the new rate.
- **Description** - A meaningful description for the new rate.
- **Rate Pattern** - Select the applicable rate pattern that applies to the normal rate code. The options include:
  - **Normal** - Obeys standard charge = price * quantity formula. For example, in the reports, the normal rate patterns are displayed as follows:

Table 2. Normal rate patterns in reports

Rate	Units	Rate Value	Charge
Storage	1,234.24	@$0.9000	1,110.82

  - **Non-Billable** - No charge applies. Included for information purposes only. For example, in the reports, the non-billable rate patterns are displayed as follows:

Table 3. Non-billable rate patterns in reports

Rate	Units	Rate Value	Charge
Monthly Storage Allowance	15,000.24		

  - **Monetary Flat Fee** - A charge levied that is not based on the volume or quantity of a metric. For example, in the reports, the monetary flat fee rate patterns are displayed as follows:

Table 4. Monetary flat fee rate patterns in reports

Rate	Units	Rate Value	Charge
Support			250.00

  - The remaining options apply to tiered rates and shifts which will be described in the related task topics.

To view screen captures of these patterns in the reports, see the IBM SmartCloud Cost Management wiki: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/IBM%20SmartCloud%20Cost%20Management/page/Welcome

8. Click **Apply** to create the rate code and return to the Rate Tables page.

Adding rates with tiers:

This topic describes how to add rates with tiers to a rate group.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. On the Rate Tables page, select the required table from the **Rate Table** dropdown menu.
3. Select the **Date** option.

   Note: The **Date** option must be selected when creating or removing rates. When the **Date** option is selected the **Effective Date** field is displayed.

4. In the **Effective Date** drop down menu, select an existing effective date or click **New Effective Date** to create a new effective date for a new set of rates.
5. If the **New Effective Date** option is selected, enter the date in the **New Date** field. Once an effective date is selected, a tree table is displayed with rate information for the date selected.
6. In the **Manage Rate Structure** tree table, expand or collapse the **Id** option to drill down through the hierarchy of different rate-related object Ids. The first node in the tree is the offering, then the rate group, then the rate followed by either rate tiers or rate shifts if they exist.
7. Right click on the required rate group and select **New Rate**. An action dialog panel is displayed with the following fields:

   Note: In this task, you want to create rates with tiers. When creating rates with tiers, the rate patterns described in this section apply.

   - **Name** - The rate name which represents the new rate.
   - **Description** - A meaningful description for the new rate .
   - **Rate Pattern** - Select the applicable rate pattern that applies to the tiered rate. The options include:
     – **Tier Individual** - Charges based on splitting quantity into multiple tiers or bands, for example, when applying a volume discount. In the reports, the tier individual rate patterns are displayed as follows in the reports:

Table 5. Tier individual rate patterns in reports

Rate	Units	Rate Value	Charge
Network traffic	901.12	@ $0.5005	451.01
– Standard Rate (<=400)	400.00	@ $0.3000	120.00
– Discount Tier 2 (<=800)	400.00	@ $0.6000	240.00
– Discount Tier 3 (<=3,000)	101.12	@ $0.9000	91.01

     – **Tier Highest** - Charges based on assigning quantity to appropriate tier or classification. For example, the tier highest rate patterns are displayed as follows in the reports:

Table 6. Tier highest rate patterns in reports

Rate	Units	Rate Value	Charge
Network traffic	2,340.24	@ $1.0000	2,340.24
– Discount Tier 3 (800 -3,000)	2,340.24	@ $1.0000	2,340.24

     – **Tier Monetary Individual (%)** - Additional monetary charge based on a percentage of another charge, for example, a tax or premium. Effective percentage depends on splitting the original charge in multiple tiers or bands. For example, the tier monetary individual (%) rate patterns are displayed as follows in the reports:

Table 7. Tier monetary individual (%) rate patterns in reports

Rate	Units	Rate Value	Charge
Support services	23.49% x $1,990.79		467.70

Table 7. Tier monetary individual (%) rate patterns in reports (continued)

Rate	Units	Rate Value	Charge
– Standard Rate (<=$200)	15.00% x $200.00		30.00
– Discount Tier 2 (<=$400)	20.00% x $200.00		40.00
– Discount Tier 3 (>$400)	25.00% x $1,590.79		397.70

– **Tier Monetary Highest (%)** - Additional monetary charge based on a percentage of another charge, for example, a tax or premium. Percentage depends on assigning the original charge to appropriate tier or classification. For example, the tier monetary highest (%) rate patterns are displayed as follows in the reports:

Table 8. Tier monetary highest (%) rate patterns in reports

Rate	Units	Rate Value	Charge
Support services	22.50% x $239.88		53.97
– Standard Rate(<=$2,000)	22.50% x $239.88		53.97

To view screen captures of these patterns in the reports, see the IBM SmartCloud Cost Management wiki: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/IBM%20SmartCloud%20Cost%20Management/page/Welcome

8. In the **Number of Tiers** menu, select the number of levels required, which is used to indicate how many rate tiers will be created automatically.
9. Click **Apply** to create the rate with tiered rates and return to the Rate Tables page.

Adding rates with shifts:

This topic describes how to add rates with rate shifts to a rate group.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. On the Rate Tables page, select the required table from the **Rate Table** dropdown menu.
3. Select the **Date** option.

   Note: The **Date** option must be selected when creating or removing rates. When the **Date** option is selected the **Effective Date** field is displayed.
4. In the **Effective Date** drop down menu, select an existing effective date or click **New Effective Date** to create a new effective date for a new set of rates.
5. If the **New Effective Date** option is selected, enter the date in the **New Date** field. Once an effective date is selected, a tree table is displayed with rate information for the date selected.
6. In the **Manage Rate Structure** tree table, expand or collapse the **Id** option to drill down through the hierarchy of different rate-related object Ids. The first

node in the tree is the offering, then the rate group, then the rate followed by either rate tiers or rate shifts if they exist.

7. Right click on the required rate group and select **New Rate**. An action dialog panel is displayed with the following fields:
   - **Name** - The rate name which represents the new rate.
   - **Description** - A meaningful description for the new rate.
   - **Rate Pattern** - Select the **Shifts** option from the dropdown menu.
     - **Shifts** - Select rate shifts to set different rates based on the time of day. For example, if a user is using computer resources at 4 a.m., you can charge the user less, rather than if the user uses these resources at 1 p.m.
8. In the **Number of Shifts** dropdown menu, select the number of levels required, which is used to indicate how many rate shifts will be created automatically.
9. Click **Apply** to create the rate with rate shifts and return to the Rate Table Maintenance page.

Importing rates

Instead of manually creating rates with a specific effective date and associating them with a new or existing rate table, you can use the Import Rates feature to import rates from an existing rate table into another rate table.

Use the Import Rates feature to do the following:

- Import all rates with a specific effective date from an existing rate table into another rate table.
- Import an offering and all its associated rates with a specific effective date from an existing rate table into another rate table.
- Import a single rate group and all its associated rates with a specific effective date from an existing rate table into another rate table.
- Import many rate groups and their associated rates with a specific effective date from an existing rate table into another rate table.
- Import a single rate with a specific effective date from an existing rate table into another rate table.
- Import many rates with a specific effective date associated with different rate groups from an existing rate table into another rate table.

Importing rates from an existing rate table:

Use the Import Rates feature to import rates with a specific effective date from an existing rate table into another rate table. You can use the same rates that were used in an existing rate table instead of manually creating them each time and associating them with the new rate table.

Before you begin

When Firebug is enabled, it may cause issues when displaying rate groups in the Import Rates dialog. It is therefore advised to disable Firebug before proceeding.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. On the **Rate Tables** panel, select the rate table that you want to populate from the **Rate Table** menu. For example, you may want to populate a new or an existing rate table with rates associated with a specific effective date from an existing rate table.

3. In the **Effective Date** field, specify a current effective date or click **New Effective Date** to create a new effective date for the new set of imported rates.
4. If the **New Effective Date** option is selected, enter the new effective date in the **New Date** field.
5. In the Manage Rates **Actions** menu, select **Import Rates**.
6. The Import Rates action dialog panel is displayed with the following properties:
   - **Rate Table** – select the rate table that you want to import the rates from.

     Note: You can also import rates into the same rate table for a new effective date.
   - **Effective Date** – specify the effective date for the rates that you want to import from the selected table. Only rates with this effective date are imported.
   - **Import Rates Tree** – This tree table allows you to import rates into an empty or existing rate table.
     a. Import rates into an empty rate table:
        – Select the top level **Offerings** node if you want to duplicate an existing table, and import all its associated offerings, rate groups, and rates.
        – Select the offering node if you want to import all rates under that offering.

          Note: When the offering node is selected, all child nodes are automatically selected.
        – Select the rate group node if you want to import all rates for that rate group.
        – Select the single rate node if you only want to import that rate.
     b. Import rates into an existing rate table:

        Attention: Rates that are marked with * in the import rates tree indicate that these rates already exist in the table the rates are being imported from. If a rate that is marked with * is selected, the existing rate in the rate table will be expired with this new rate. A label is also displayed at the end of the import rates tree showing the number of new and existing rates selected for import.
        – Select the top level **Offerings** node if you want to duplicate an existing table, and import all its associated offerings, rate groups, and rates.
        – Select the offering node if you want to import all rates under that offering.

          Note: When the offering node is selected, all child nodes are automatically selected.
        – Select the rate group node if you want to import all rates for that rate group.
        – Select the single rate node if you only want to import that rate.
   - Click **Import** or **Cancel** to cancel the import and return to the **Rate Tables** panel.
   - If the import option is selected, all rates with the specified effective date are imported and added to the rate table.
   - Click **Yes** to import or **No** to cancel.

Results

A message is displayed indicating that the rates were imported correctly. Click **OK** to return to the **Rate Table Maintenance** panel.

Moving rates between rate groups

Moving rates between rate groups avoids the requirement to create new rates each time you want to assign a rate to a rate group.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. Expand the required offering and locate the rate group that contains the rate or rates that you want to move.
3. Click and drag the rate to the required rate group, either in the same offering or in another offering.

   Note: When the drop box is green, it means that you can move the rate to that location, for example, to another rate group. If the drop box is red, it means that you are not allowed to move the rate to that location, for example, in the same rate group.

Results

The rate is moved to another rate group, either in the same offering or another offering, and it is saved automatically.

Note: After completing this task, the rate reordering is not updated by default in the Rate Tables page. You must refresh the Rate Tables page to see the new rate sequence.

Changing the rate sequence

The sequence that the rates are displayed in the tree-table is the sequence that they are displayed in the reports. You can sequence the rate in any order within the same rate group.

Procedure

1. In Administration Console, click **Administration** > **Rate Groups**.
2. Expand the required offering and locate the rate group whose rates sequence you want to change.
3. Click the rate and drag it to the new location in the rate group.

   Note: When the drop box is green, it means that you can move the rate to that location, for example, either before or after an existing rate in the rate group. If the drop box is red, it means that you are not allowed to move the rate to that location, for example, if you drag a rate onto another rate.

Results

The rate sequence is changed, the screen is refreshed and the new rate ordering is displayed.

Note: After completing this task, the rate reordering is not updated by default in the Rate Tables page. You must refresh the Rate Tables page to see the new rate

sequence.

Modifying rates

This topic describes how to modify rates.

Before you begin

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. On the Rate Tables page, select the required table from the **Rate Table** dropdown menu.
3. Select the **Date** or **Date Range** option.

   Note: When modifying rates, you can select either the **Date** or **Date Range** option. Select the **Date** option when looking for rates for a specific date. The **Date Range** option is used to search for rates for a specific date range.
4. Depending on whether the **Date** or **Date Range** option is selected, complete the following fields:

   Effective Date - When the Date option is selected.
   A list that contains all of the effective dates for the different sets of rates that exist in the selected rate table. The list also contains a **New Effective Date** option used to create a new effective date for a new set of rates. When looking for rates for a specific date, select the **Date** radio button option and an existing effective date in the **Effective Date** field. Once the date is entered, a tree table is displayed with rate information for the date selected.

   Effective From - When the Date Range option is selected.
   A list that contains all of the effective dates for the different sets of rates that exist in the selected rate table. When looking for rates for a date range, select the **Date Range** radio button and enter an existing effective date in the **Effective From** field. This field indicates the lowest effective date in a date range, when searching for rates to display in the main tree table. This field is used in conjunction with the **Effective To** field to create the date range search criteria. Once the **Effective From** and **Effective To** dates are entered, a tree table is displayed with rate information for the date range selected.

   Effective To
   A list that contains all of the expiry dates for the different sets of rates that exist in the selected rate table. It is used to indicate that rates with an effective date after this date are to be excluded from the date range search. This field is used in conjunction with the **Effective From** field to create the date range search criteria. Once the **Effective From** and **Effective To** dates are entered, a tree table is displayed with rate information for the date range selected.

   New Date
   When the **New Effective Date** is selected in the **Effective From** menu, use the **New Date** field to specify a new date.
5. Once an effective date or effective date range is selected, a tree table is displayed with rate information for the date or date range selected.
6. Expand or collapse the **Id** option to drill down through the hierarchy of different rate-related object Ids. The first node in the tree is offering, then the

rate group, then the rate followed by either rate tiers or rate shifts if they exist. You can do inline editing on the main tree table for the following fields in the table:

Description
A meaningful name for the rate-related object contained in the tree table row.

Effective Date
The **Effective Date** shows when the rate becomes effective. When a rate becomes effective it is used when processing metering data for records generated on or later than that date and before the expiry date of the rate.

Expiry Date
The **Expiry Date** shows the date when the rate expires. Metering records which include this rate and are generated after this date will no longer use this rate.

Threshold
Tiered rates allow for the specification of a threshold or cutoff value for each tier. The threshold is the unit or monetary boundary value for the tier and anything with a smaller threshold value is charged at that rate.

Percentage
Tiered rates created with a rate pattern value of **Tier Monetary Individual (%)** or **Tier Monetary Highest (%)** define the percentage of the overall rated monetary value that is charged.

Rate Value
The rate value represents the per unit value that is specified for the rate, rate tier, or rate shift. The rate value is the amount charged for the consumption of the resource represented by this rate. The value is multiplied by the resource amount contained in matching CSR or CSR+ file. For example, if the rate value is 25 and a matching resource file contains a value of 5 hours, then the total charge is $125.

Note: The rate value corresponds to the specified rate:
- $25 is input as 25
- $1.25 is input as 1.25
- Negative values are preceded by a minus (for example, –1)

7. Right-click on the rate code you want to edit and select the **Edit Properties** option.
   - **Edit Properties** - this menu option opens an action dialog panel with the following advanced rate properties:
     - **Detail Description** - This field is user-specified. Type a description for the rate that you want to use in the custom reports.
     - **Comments** - If required, enter comments for the rate.
     - **Rate Pattern** - A read only field containing the rate pattern which was used when creating the rate and its child rate tiers or rate shifts. The values displayed include:
       - **Normal** - Obeys standard charge = price * quantity formula.
       - **Non-Billable** - No charge applies. Included for information purposes only.
       - **Monetary Flat Fee** - A charge levied that is not based on the volume or quantity of a metric.

- **Tier Individual** - Charges based on splitting quantity into multiple tiers or bands, for example, when applying a volume discount.
- **Tier Highest** - Charges based on assigning quantity to appropriate tier or classification.
- **Tier Monetary Individual (%)** - Additional monetary charge based on a percentage of another charge, for example, a tax or premium. Effective percentage depends on splitting the original charge in multiple tiers or bands.
- **Tier Monetary Highest (%)** - Additional monetary charge based on a percentage of another charge, for example, a tax or premium. Percentage depends on assigning the original charge to appropriate tier or classification.
- **Shifts** - Rate shifts allow you to set different rates based on the time of day. For example, if a user is using a computers resources at 4 a.m., you can charge the user less, rather than if the user uses these resources at 1 p.m.

– **Do not adjust for Zero cost** - Select this check box if you do not want the associated rate included in zero cost calculations.
– **CPU Value** - Select this check box to normalize CPU usage for this rate.
– **Average** - Select this check box to average all values corresponding to this rate.
– **Report Flag 1 and 2** - The use of these check boxes is user-specified. Select these check boxes to type a one-character value that you can use in custom reports.
– **Resource Conversion** - You can adjust the total resource units value in reports using the following conversion factors:
  - **Default** - No conversion is performed.
  - **Divide By or Multiply By** - The total resource units are divided or multiplied by a set conversion factor, for example, Divide By 1000, Multiply By 60, .
  - **Multiply By Conversion Factor** - The total resource units are multiplied by the factor in the **Rate Conversion Factor** field.
– **Rate Conversion Factor** - This field is available only when **Multiply By Conversion Factor** is selected in the **Resource Conversion** field. Type the number that you want to multiply the total resource units for the rate by. This factor can be up to 16 digits, including a decimal.
– **Rate is per thousand** - Select this check box to change the rate in reports from per resource unit to per thousand units.
– **Use 4 decimals for rate** - This option determines the number of decimal digits that are displayed in the rate value in reports. If this check box is selected, the rate value includes four decimal digits. If this check box is not selected, the rate value includes eight decimal digits.
– **Resource Decimals** - Select the number of decimal digits that are displayed in the resource units value in reports. For example, 0=99, 2=99.99, 4=99.9999. The default is two decimal digits.

Click **Save** to update the rate with the advanced properties entered or **Cancel** to return to the Rate Tables panel.

Deleting rates and rate group rates

This topic describes how to delete rates. This procedure applies to normal rates, rates with tiers, and rates with shifts

About this task

You have two options when removing rates. You can either remove all rates associated with a rate group, or remove a specific rate.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. Select the required rate table from the **Rate Table** drop down menu.
3. Use the **Date** option to locate the rate or rate group rates that you want to remove.
4. Expand or collapse the **Id** option to drill down through the hierarchy of different rate-related object Ids. The first node in the tree is the offering, then the rate group, then the rate followed by either rate tiers or rate shifts if they exist.
5. To remove all rate group rates:
   - Right click on the rate group row on the Rate Tables page.
   - Select the **Remove Rate Group Rates** option.
   - In the deletion confirmation pop up panel, select **Yes** to delete the rate group rates or **No** to cancel the deletion.
6. To remove a specific rate:
   - Right click on the required rate on the Rate Tables page, and select **Remove Rate**.
   - In the deletion confirmation pop up panel, select **Yes** to delete the rate group rates or **No** to cancel the deletion.

   Note: When deleting a specific rate or all rate group rates, they are merged backwards by default. This means that the first rate or set of preceding rates that match those being removed will have their expiry dates updated to the expiry dates of the rate or rates being removed. This is required to avoid any gaps in rate effective dates.

Defining rate templates

A Rate Template is an XML file which represents the structure and types of rate templates which can be created in SmartCloud Cost Management. The rate template is used to define the base structure of the rates and rate groups which represent the offering.

The rate template ensures that users can create multidimensional rates and allows users to quickly generate sets of related rate groups and rates for a rate table. Some examples of multidimensional rate templates that can be represented by and generated from the rate template include:

Table 9. IaaS style Offering

Quality of service	Software stack	Size
Gold, Silver, Bronze	Windows, Linux	Large, Medium, Small

Table 10. Virtual Server Service style Offering

Hypervisor	Resources
VMware, LPAR, KVM, Xen	CPU, Storage, Memory, Server

Some examples of the dimensions in the sample IBM Cloud Orchestrator rate templates that can be represented by and generated from the rate template include:

Table 11. License Charges Offering

Licenses	Software
`Licenses`	WebSphere Application Server 8, WebSphere MQ 7, DB2 10, IBM AIX® Version 7, Windows Server 2012

Table 12. Infrastructure Charges Offering

Infrastructure	Architecture	Resource
`Infrastructure`	Power®, x86	CPU Hour, Memory GB Hour, Storage GB Hour

Table 13. Hosting Charges Offering

Network Zones	Hypervisor	Resource
`Dev Test, Production`	LPAR, KVM	Base OS

Table 14. Hosting Charges with VM Sizes Offering

Network Zones	Hypervisor	Resource
`Dev Test, Production`	LPAR, KVM	Tiny VM, Small VM, Medium VM, Large VM, Extra Large VM

These examples are all available in the sample `<SCCM_install_dir>/offerings/offering.xml`. The IBM Cloud Orchestrator examples provided are used in the `SampleOpenStack.xml` job file. See the related topic for more information about see this job file.

Resource elements for the rate template

Use this topic to understand the resource elements that are used in the rate template.

Offering resource element

Note: All the property elements described in the following tables are mandatory.

Table 15. Properties of the Offering resource element

Property name	Generic type	Description
`id`	String	Identifier for one of the offerings specified in the offering template file.
`shortCode`	String	The `shortCode` identifier for the offering. This `shortCode` is used as a portion of the `RateGroup` and `RateCode` identifiers when they are created.

Table 15. Properties of the Offering resource element (continued)

Property name	Generic type	Description
description	String	A detailed description of the offering.
rateDimensions	List <RateDimension>	All rate dimensions associated with this offering. At least 1 dimension must be specified for an offering.

RateDimension resource element

Table 16. Properties of the RateDimension resource element

Property name	Generic type	Description
description	String	A detailed description of the rate dimension.
dimensionType	Enum	The enum value indicates if whether the dimension is part of a RateGroup, RateCode or RateShift. The accepted values are one of the following: • RATEGROUP • RATECODE • RATESHIFT In a list of RateDimensions, at least 1 dimensionType must be RATECODE.
order	Integer	Indicates the order or position of the rate dimension when generating the RateGroup or RateCode.
elementLength	Integer	Element length indicates what the rateDimension elementId lengths are. This element is required to identify what portion of the RateGroup or RateCode the element belongs to.
rateDimension Elements	List <RateDimension Element>	All rate dimension elements associated with this rateDimension. At least 1 dimension element must be specified for a rateDimension.

RateDimensionElement resource element

Table 17. Properties of the RateDimensionElement resource element

Property name	Generic type	Description
shortCode	String	The shortCode identifier for the dimension element. This shortCode element is used as a portion of the RateGroup and RateCode identifiers when they are created.

Table 17. Properties of the RateDimensionElement resource element (continued)

Property name	Generic type	Description
description	String	Description is the name of the element and is part of the RateGroup or RateCode description.

Modifying the sample rate templates

Use the sample rate templates provided in this topic as an example of how to create new rate templates and modify your existing rate template.

CAUTION:
You must copy the sample xmls in exactly the same format as they are displayed in this topic. For example, incorrect spacing may result in issues. See the *Troubleshooting Guide* **for common issues that may occur if you fail to copy the xml correctly.**

The sample offerings template described in this topic must remain within the following tags:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <ns2:offerings xmlns:ns2="http://www.ibm.com/xmlns/prod/tuam/1.0">
 <!--
 Offerings described below
  -->
 </ns2:offerings>
```

The RDP offering template

```
<offering id="RDP" description="Virtual Server Service" shortCode="RDP">
 <rateDimensions>
  <rateDimension description="Hypervisor">
   <dimensionType>RATEGROUP</dimensionType>
   <order>0</order>
   <elementLength>1</elementLength>
   <rateDimensionElements>
    <rateDimensionElement description="VMware" shortCode="V"/>
    <rateDimensionElement description="LPAR" shortCode="L"/>
    <rateDimensionElement description="KVM" shortCode="K"/>
    <rateDimensionElement description="HyperV" shortCode="H"/>
    <rateDimensionElement description="zVM" shortCode="Z"/>
   </rateDimensionElements>
  </rateDimension>
  <rateDimension description="Component">
   <dimensionType>RATECODE</dimensionType>
   <order>1</order>
   <elementLength>3</elementLength>
   <rateDimensionElements>
    <rateDimensionElement description="Server Hour" shortCode="SRV"/>
    <rateDimensionElement description="CPU Hour" shortCode="CPU"/>
    <rateDimensionElement description="Memory GB Hour" shortCode="MEM"/>
    <rateDimensionElement description="Storage GB Hour" shortCode="STR"/>
   </rateDimensionElements>
  </rateDimension>
 </rateDimensions>
</offering>
```

The IaaS offering template

```
<offering id="IaaS" description="IaaS Offering" shortCode="IAAS">
  <rateDimensions>
   <rateDimension description="Quality Of Service">
```

```
    <dimensionType>RATEGROUP</dimensionType>
    <order>0</order>
    <elementLength>1</elementLength>
    <rateDimensionElements>
     <rateDimensionElement description="Gold" shortCode="G"/>
     <rateDimensionElement description="Silver" shortCode="S"/>
     <rateDimensionElement description="Bronze" shortCode="B"/>
    </rateDimensionElements>
  </rateDimension>
   <rateDimension description="Software Stack">
    <dimensionType>RATEGROUP</dimensionType>
    <order>1</order>
    <elementLength>1</elementLength>
    <rateDimensionElements>
     <rateDimensionElement description="Linux" shortCode="L"/>
     <rateDimensionElement description="Windows" shortCode="W"/>
   </rateDimensionElements>
  </rateDimension>
   <rateDimension description="T-Shirt Size">
    <dimensionType>RATECODE</dimensionType>
    <order>2</order>
    <elementLength>1</elementLength>
    <rateDimensionElements>
     <rateDimensionElement description="Large" shortCode="L"/>
     <rateDimensionElement description="Medium" shortCode="M"/>
     <rateDimensionElement description="Small" shortCode="S"/>
   </rateDimensionElements>
  </rateDimension>
 </rateDimensions>
</offering>
```

Importing the rate template

Use this topic to import a rate template.

About this task

Note: You can import a rate template only when the **Date** option is selected.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. Select the **Date** option.

   Note: The **Date** option must be selected when importing or removing rate templates. When the **Date** option is selected the **Effective Date** field is displayed.
3. In the **Effective Date** drop down menu, select an existing effective date or click **New Effective Date** to create a new effective date for the set of rates that are specified in the rate template.
4. If the **New Effective Date** option is selected, enter the date in the **New Date** field for the set of rates that are specified in the rate template.
5. In the **Manage Rates** tree table toolbar, click the **Actions** menu option. This menu contains some options that are specific to SmartCloud Cost Management.
6. Select the **Import Rate Template** option from the **Actions** menu.
7. In the **Rate Template** menu, select either the sample rate template that is supplied as part of the product installation or a modified sample XML file that you have created.
8. Click the **Apply** to import the offering. Rate groups and rates that are defined in the template are then created for the specified effective date.

Deleting the rate template

This topic describes how to delete a rate template.

Before you begin

Note: You can only remove a rate template when the **Date** option is selected.

Procedure

1. In Administration Console, click **Administration** > **Rate Tables**.
2. Select the **Date** option.

   Note: The **Date** option must be selected when importing or removing rate templates. When the **Date** option is selected the **Effective Date** field is displayed.
3. In the **Manage Rates** tree table toolbar, click the **Actions** menu option. This menu contains some options that are specific to SmartCloud Cost Management.
4. Select the **Remove Rate Template** option from the **Actions** menu.
5. In **Rate Template** menu select the rate template that you want to remove.
6. The **Merge Backward** option is enabled by default so that the first set of preceding rates that match those being removed will have their expiry dates updated to the expiry dates of the rates being removed. This is required to avoid any gaps in rate effective dates.
7. Click **Apply** to remove the rate template.

Defining the calendar

SmartCloud Cost Management includes a default calendar with the standard 12 monthly periods (the first day of the month to the last day) by year. Organizations using standard monthly/yearly periods do not need to change the calendar. If you do not want to run billings using standard monthly/yearly periods, you can change the period start and end dates and optionally include 13 periods per year.

Calendar considerations

Calendar data is stored in the `CIMSCalendar` table. The maximum entries SmartCloud Cost Management reads from the `CIMSCalendar` table is 52. Therefore, you must delete periods from previous years as they pass.

The `CIMSCalendar` table must have the current and previous periods defined.

Setting up the calendar

This topic describes how to create a calendar year with periods.

Procedure

1. In Administration Console click **Administration** > **Calendar**.
2. On the Calendar page, select the year that you want in the **Year** dropdown list and complete the following:

   Year Select the appropriate year if it is not shown.

   New Year
   Click to add a year to the calendar.

- If you want to create a 12 period calendar, make sure that the **Use 13 Periods** is not selected. Enter a year value. Click **Create** to proceed or **Canel** to return to the Calendar page.
- If you want to create a 13 period calendar, make sure that **Use 13 Periods** is selected. Enter a Start Date for the period, use the View Calendar drop down icon to select a start date, or accept the default date. Click **Create** to proceed or **Cancel** to return to the Calendar page.

  Note: The default 13 period year consists of 28 day period and ends on the 29th of December. You must manually adjust the last period to include the remaining days of the year.

Delete Year
Click to delete the currently shown year.

Use 13 Periods
Select this option if you want to change from the default 12 periods calendar to 13 periods calendar.

Calendar Periods
Shows 12 periods for the year, or 13 periods if **Use 13 Periods** is selected. Select the period that you want to edit.

Period Shows the id of the period currently selected.

Start Date
Shows the date the currently selected period started on. Type a start date for the period, use the View Calendar dropdown to select a start date, or accept the default date.

Note: When modifying the start date, it cannot have the same date or a date greater than the end date.

End Date
Shows the date the currently selected period ends on. Type an end date for the period, use the View Calendar dropdown to select an end date, or accept the default date.

Note: When modifying the end date, it cannot have the same date or a date less than the start date. There cannot be more than 31 days between a single period start and end date.

Apply Click to apply the settings.

Set From Period 1
Click to have all periods processed sequentially from the first period.

Reset to Defaults
Click to reset any modified periods to their original values, if these changes have not been previously committed.

Defining configuration options

This section describes how to set SmartCloud Cost Management configuration settings.

You can configure the following SmartCloud Cost Management configuration settings:

- **Driver options.** Used to add the required JDBC drivers for the SmartCloud Cost Management database. JDBC is an API specification for connecting programs written in Java to the data in a wide range of databases.
- **Logging options.** Used to set the trace and log file settings for SmartCloud Cost Management. Trace files contain trace messages that show SmartCloud Cost Management execution statements. Log files contain informational, warning, and severe messages related to SmartCloud Cost Management activities such as installation and processing data.
- **Organization information.** Used to type the name and address of your organization. This information is displayed on the standard invoices that are provided with SmartCloud Cost Management.
- **Processing options.** Used to set the processing settings for SmartCloud Cost Management.
- **Reporting options.** Used to set the reporting settings for SmartCloud Cost Management.

Adding a JDBC driver

The appropriate JDBC driver must be available for the SmartCloud Cost Management database and any databases from which data is collected by SmartCloud Cost Management Data Collectors. JDBC is an application program interface (API) specification for connecting programs written in Java to the data in a wide range of databases. The appropriate JDBC drivers must be available on the server running SmartCloud Cost Management.

About this task

The database used to store SmartCloud Cost Management data must use the following driver:

- For DB2 for Linux db2jcc.jar and db2jcc_license_cu.jar (license JAR file), where the version is appropriate for the database to be used in the data source for SmartCloud Cost Management.

You can use other drivers for databases used by SmartCloud Cost Management Data Collectors.

Procedure

To add a new JDBC driver, copy the jar file to `<SCCM_install_dir>/wlp/usr/servers/sccm/dbLibs`.

Note: Restarting the server is not required.

Setting logging options

This topic describes how to set options for logging such as maximum log or trace file size, maximum number of archive files, and so forth.

About this task

Procedure

Open the `logging.properties` file which is located in `<SCCM_install_dir>/config` and edit the following parameters. After making the changes in this file, you must restart the application server.

Table 18. Logging options

Parameter	Description
com.ibm.tivoli.tuam.logger.MessageFileHandler.limit **com.ibm.tivoli.tuam.logger.TraceFileHandler.limit**	Type the size for the trace or log file in bytes. SmartCloud Cost Management writes messages to this file until the data exceeds the file size. When the data exceeds the file size, SmartCloud Cost Management archives the current file, creates a new file, and continues to add messages to the new file. The number of archive files is set using the **com.ibm.tivoli.tuam.logger.MessageFileHandler.count** parameter.
com.ibm.tivoli.tuam.logger.MessageFileHandler.count **com.ibm.tivoli.tuam.logger.TraceFileHandler.count**	Type the maximum number of archive files that you want to use for trace and log files. When the data in a trace or log file exceeds the file size, SmartCloud Cost Management archives the current file, creates a new file, and continues to add messages to the new file. If the number of archive files exceeds the maximum number of archive files, the oldest archive file is removed.
com.ibm.tivoli.tuam.logger.MessageFileHandler.level **com.ibm.tivoli.tuam.logger.TraceFileHandler.level**	• For trace messages, select one of the following: **FINE**, **FINER**, or **FINEST**, where **FINEST** provides the most detailed messages. • For log messages, select one of the following: **SEVERE**, **WARNING**, or **INFO**, where **SEVERE** provides only severe messages; **WARNING** provides severe and warning messages; and **INFO** provides severe, warning, and information messages.

Adding organization information

Your organization information is displayed on the standard invoices that are provided with SmartCloud Cost Management.

Procedure

Open the `config.properties` file which is located in `<SCCM_install_dir>/config` and edit the following parameters:

Table 19. Organization information

Parameter	Description
com.ibm.tivoli.tuam.config.ORGNAME	Type the name of your organization as you want it to appear on invoices and any other reports.

Table 19. Organization information (continued)

Parameter	Description
com.ibm.tivoli.tuam.config.ADDRESSLINE1 -- Address Line 1 through Address Line 4	Type the address for your organization as you want it to appear on invoices and any other reports.

Note: Because this is a Java properties file, if you need to include any non-ASCII (Unicode) characters, these must be converted using the `native2ascii` tool. For example, the string `Siège de l'entreprise` would become `Si\u00e8ge de l'entreprise`. This tool is available as part of the Java Virtual Machine installed with SmartCloud Cost Management.

Setting processing options

This topic describes how to set options for data processing such as the processes folder path, the job files path, and so forth.

Procedure

Open the `config.properties` file which is located in `<SCCM_install_dir>/config` and edit the following parameters:

Table 20. Processing options

Parameter	Description
com.ibm.tivoli.tuam.config.CollectorLogFilesPath	Type the path for the SmartCloud Cost Management collector log files, or accept the default path. Collector log files are usage metering files that are created by some SmartCloud Cost Management Data Collectors. These usage files are processed by SmartCloud Cost Management.
com.ibm.tivoli.tuam.config.JobFilesPath	Type the path for the SmartCloud Cost Management job files, or accept the default path. A job file is an XML file that defines the process of collecting data and loading it into a SmartCloud Cost Management database.
com.ibm.tivoli.tuam.config.JobLogFilesPath	Type the path for the SmartCloud Cost Management job log files, or accept the default path. A job log 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 or failure occurred.
com.ibm.tivoli.tuam.config.JobSampleFilesPath	Type the path for the SmartCloud Cost Management sample job files, or accept the default path. Sample job files are provided with SmartCloud Cost Management in the *<SCCM_install_dir>*`/samples/jobfiles` directory.
com.ibm.tivoli.tuam.config.ProcessDefinitionPath	Type the path for the SmartCloud Cost Management process definitions, or accept the default path. A process definition is a folder that contains the files required to process usage data from a particular source such as a database, operating system, or application.

Note: The paths can contain the property, `${sccm.install.dir}`, which is the directory where SmartCloud Cost Management is installed.

Setting reporting options

This topic describes how to set options for reporting, such as the level of account codes that are shown in the reports.

Procedure

Open the `config.properties` file which is located in `<SCCM_install_dir>/config` and edit the following parameters:

Table 21. Reporting options

Parameter	Description
com.ibm.tivoli.tuam.config.ACCOUNTPARAMETERSELECTIONLEVEL	This setting determines the level of account codes that are displayed in the **Starting Account Code** and **Ending Account Code** parameter lists when running the reports. For example, if you type 1, only the first level account codes are displayed. However, if you type 3, the first, second, and third level account codes are displayed.

Stopping and starting the Application Server

The Application Server starts automatically after it has been installed. You can manually stop the server before beginning certain configuration tasks or as needed.

1. On the command-line interface, change to the `<SCCM_install_dir>/bin` directory.
2. **Stop** the server using the following command:
   - For Linux - run the following command: `./stopServer.sh sccm`
3. Wait a moment for the server to completely shut down.
4. **Start** the server:
   - For Linux - run the following command: `./startServer.sh sccm`

Chapter 4. Administering data processing

This section contains information about the data collection and processing tasks that you can perform with the product.

Data processing architecture

The SmartCloud Cost Management data processing architecture consists of the programs, directories, and files used to process resource usage data.

Job Runner

The SmartCloud Cost Management Job Runner application runs jobs that are defined in a job file. Each job can run one or more SmartCloud Cost Management Data Collectors.

You can run job files using Job Runner in either of the following ways:

- **In batch (recommended).** Using the SmartCloud Cost Management console application, Job Runner, SmartCloud Cost Management Data Collectors can be run in batch mode on a defined schedule (daily, monthly, and so on) to convert and usage metering data created by your system into CSR or CSR+ files. After the CSR or CSR+ files are created, the data in the files is processed and the output is loaded into the database using the SmartCloud Cost Management Processing Engine.

Job files

A job file is an XML file that defines the data collection and processing. 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 SmartCloud Cost Management components required to process the data and load it into a database.

SmartCloud Cost Management includes sample job files that you can modify for your organization. These files are in the *<SCCM_install_dir>*\samples\jobfiles directory. For example, assume that you want to collect AIX Advanced Accounting data from several servers so that you can report on and bill for the resources consumed. the \samples\jobfiles includes a SampleAIXAA.xml job file that you can rename and modify for your AIX Advance Accounting set up.

Note: If you modify any sample job file, rename the file. Otherwise, the file will be overwritten when you install a new version of SmartCloud Cost Management Data Collectors. The sample job files are intended to be run on a nightly basis to run one or multiple data collectors. However, you can schedule Job Runner to run job files on any schedule.

Running sample job files

You can run sample job files using the RunSamples.bat or RunSamples.sh script in the *<SCCM_install_dir>*\bin directory.

© Copyright IBM Corp. 2013, 2015

The RunSamples scripts are used to create sample data that can be viewed in reports. Once SmartCloud Cost Management is being used with live production data, the sample data should be deleted using the Tracking database loads page. See related topic for more information.

The RunSamples scripts call most job files using a-date parameter. The -date parameter determines the start date and end date of the output data. For example, if the -date parameter is set to yyyy0601 (where yyyy is the year), the start and end dates for the data will be June 01, yyyy. For those job files that are not called with the -date parameter, the date defined in the job file is used to determine the start and end dates.

The following are the commands for running the RunSamples script in a Linux environment:

- **Linux:** Using a shell command, type: <*SCCM_install_dir*>/bin/RunSamples.sh

Job file XML schema

The job files use an XML schema, <*SCCM_install_dir*>\config\schemas\TUAMJobs.xsd. This schema defines and validates the structure of the job file(s).

Note: It is important that you do not modify the job file schema.

The following definitions are included in the schema:

- 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.

Collection files

Collection files are used by some SmartCloud Cost Management data collectors to collect and convert usage data produced by an application. Collection files are stored in the <*SCCM_install_dir*>\collectors directory.

Note: If you modify a file or script in the collectors folder, it is important that you rename the file. Otherwise, the file will be overwritten when you upgrade to a new version of SmartCloud Cost Management.

Other Files

Depending on the collector, the collector subfolder might contain any of the following files: executable, XML, and other files used to install and configure the collector.

Log files

SmartCloud Cost Management provides a variety of log files that provide informational and troubleshooting data. If you require assistance from IBM Software Support, you might be asked to provide one or more of these files.

Message and trace log files

Message and trace log files provide results for SmartCloud Cost Management .

The following message and trace log files are in the <*SCCM_install_dir*>/logs/server directory. In general, trace files are most useful for technical support while message files provide more user-friendly information that is translated into multiple languages.

- **trace**<***archive index number***>**.log.**<***process index number***>

  This log file contains trace messages generated by the SmartCloud Cost Management application.

- **message**<***archive index number***>**.log.**<***process index number***>

  This log file contains information, error, and warning messages generated by the SmartCloud Cost Management application.

The archive index number specifies the chronology of the archived files. The higher the archive index number, the older the file. Therefore, zero (0) is the current file. All files from one (1) on are archived files. You set the number of archived files that you want to retain in the logging.properties file. For more information about this file, see the related topic about setting logging options.. The default is 11.

The process index number is used by the Java Virtual Machine (JVM). Multiple virtual machines running concurrently cannot access the same trace or log files. Therefore, separate trace and log files are created for each virtual machine. The process index number is not configurable.

Configuring message and trace logs

You can set the maximum log file size, the level of detail that you want provided in the files, and the number of files that you want to archive in the <SCCM_install_dir>/config/logging.properties file. After making the changes in this file, the application server must be restarted.

Job log files

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 or failure occurred. You can view job log files in Administration Console.

Note: The SmartCloud Cost Management trace and message log files in the <*SCCM_install_dir*>/logs/server directory contain additional details about the jobs that are run.

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 Administration Console before scheduling the job to run in batch.

Setting the job log file directory path

The path to the job log files is defined in the `config.properties` file. For more information about this file, see the related topic about setting processing options.

Defining the log file output type

You can produce log data in a text file, an XML file, or both. By default, both a text file and an XML file are produced.

If want to produce one file type only, include the attribute `joblogWriteToTextFile="false"` or `joblogWriteToXMLFile="false"` in the job file.

If you want to view the log files in Administration Console, the job log files must be in XML format.

Text log files are named `yyyyMMdd_hhMMss.txt`. XML log files are named `yyyyMMdd_hhMMss.xml`. Where `yyyyMMdd_hhMMss` is the job execution time

Sending log files through email

You can choose to have output log files sent using email to a recipient or recipients. To send log files via email, set the appropriate SMTP attributes in the job file.

The log files that are attached to an email message are `.txt` files, regardless of the value for the attribute `joblogWriteToTextFile`. If both the `joblogWriteToTextFile` and `joblogWriteToXMLFile` attributes are set to `"false"`, the email message is empty and no log file is attached.

Note: If the `smtpSendJobLog` is set to `"true"`, and the email message cannot be delivered, a warning message is logged in the SmartCloud Cost Management message and trace files and in the command console.

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.

Embedded Web Application Server log files

WebSphere Liberty log files provide results for various functions related to SmartCloud Cost Management.

These log files are in the `<SCCM_install_dir>/wlp/usr/servers/sccm/logs` directory. The most important logs in this directory are:

`console.log`
: This log captures the application server system messages and **`System.out`** output generated by the SmartCloud Cost Management application.

trace.log
This WebSphere JVM log captures trace output generated by the application server.

Installation log files

A log file is created each time that SmartCloud Cost Management, SmartCloud Cost Management Windows Process Collector, or DB2 Runtime Client are installed or uninstalled. This log file provides results for each step in the install or uninstall process. If a warning or failure occurs during the installation or unistallation, the file indicates at which point the warning or failure occurred.

The install log files for SmartCloud Cost Management Windows Process Collector are stored in:

```
%PROGRAMFILES%\IBM\tivoli\common\AUC\logs\install
```

The install log files for SmartCloud Cost Management are stored in:

```
<SCCM_install_dir>/logs/install.log
```

Processing programs

The following processing programs are required: Integrator or Acct; Scan; Bill; and DBLoad. All other programs are optional.

Integrator

The Integrator program enables you to modify input data provided in a variety of formats (including CSR or CSR+ files). Integrator includes the functions provided by the Acct program (account code conversion, shift determination, and date selection) plus many other features such as adding an identifier or resource to a record, converting an identifier or resource to another value, or renaming identifiers and resources.

Scan

The Scan program performs the following tasks:

- Verifies that the feed subdirectory or subdirectories in a process definition directory contain a CSR or 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 .
- Concatenates the CSR or CSR+ files produced by data collectors of the same type from multiple servers into one file.
- Outputs a CSR or CSR+ file (whether from one server or a concatenated file from multiple servers) to the collector's process definition directory. The default file name for the CSR file is CurrentCSR.txt.

Note: 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 must move the CSR or CSR+ file contained in the feed subdirectory to the collector's process definition directory.

Acct

Note: Acct was formerly CIMSAcct in previous releases of SmartCloud Cost Management. If you have upgraded to SmartCloud Cost Management 2.1.0.5 from

a previous release and are using the CIMSAcct program it is recommended that you phase out the use of Acct and begin using the more powerful Integrator program.

The Acct program performs account code conversion, shift determination, date selection, and identifier extraction on the usage data provided in CSR or CSR+ files, and produces CSR+ file that is used as input into the Bill program.

Bill

The Bill program processes the CSR or CSR+ file from Integrator or Acct and performs shift processing, CPU normalization, and include/exclude processing and creates the Ident, Detail, and Summary files. These files contain the billing information used to generate invoices and reports. Bill also applies the rates to cost the data.

DBLoad

The DBLoad program loads the output files from Integrator, Acct, and Bill into the SmartCloud Cost Management database.

WaitFile

The WaitFile program directs Job Runner to wait for one or more files before continuing processing.

File Transfer

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 SmartCloud Cost Management application server.

Cleanup

The Cleanup program deletes files with file names containing the date in yyyymmdd format in the collector's process definition directory or any other directory that you specify (for example, the directory 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.

Process definitions

Process definitions are directories that contain the files required to process usage data from a particular source such as a database, operating system, or application. Process definition directories are also used to store the CSR or CSR+ files that are generated from the usage data.

A separate process definition directory is required for each application that you collect data from. If a process definition directory does not exist for the collector, Job Runner can create a directory using the process ID defined in the job file as the directory name.

About the process directory

The processes directory is located in the following directory when SmartCloud Cost Management is installed:

- <*SCCM_install_dir*>\samples\processes

It is recommended that you copy it to a location where you keep data that is backed up.

Each time that you upgrade to a new release of SmartCloud Cost Management, a new \samples\processes directory is installed. You can then copy or move any new process definition directories that you want from the \samples\processes directory to your processes directory. Each process definition directory contains the files and subdirectories described in the following sections.

Note: The path to your processes directory must be defined in the <*SCCM_install_dir*>\config\config.properties file.

Feed subdirectory

A feed subdirectory is automatically created in the process definition directory 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 subdirectory 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).

Each feed subfolder is used to store CSR or CSR+ files from the feed of the same name. The CSR or CSR+ file name contains a date in yyyymmdd format. (Note that although the feed subdirectory is created in the Transactions process definition directory, it is not used. CSR files created by the Transactions collector are placed directly in the process definition folder.) The Scan program processes and concatenates the CSR or CSR+ files in the feed subdirectories. The resulting output file is placed directly in the process definition directory.

Note: To prevent data processing errors, the process definition directory should not contain subdirectories other than feed directories and feed directories should not contain files other than CSR files.

Additional Processing Files

Each process definition folder contains additional processing files that are used internally by SmartCloud Cost Managementr.

Date keywords

SmartCloud Cost Management includes keywords that specify the dates for the data that you want to collect. You can also provide date literals. You can provide these date keywords or literals when you are running a job file or within the job file itself.

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

- CURWEK (Selects data produced in the current week [Sun–Sat].)
- CURMON (Selects data produced in the current month.)
- Date in yyyymmdd format (Selects data produced on a specified date.)
- Date in yyyypp format (Selects data produced in a specified period as defined by the Calendar table. This is used by the Transactions collector only.)
- Date range in yyyymmdd yyyymmdd format (selects data produced in a specified date range.)

Data processing overview

SmartCloud Cost Management processes and applies business rules to resource usage data from any application, system, or operating system on any platform. The primary method for loading this data into SmartCloud Cost Management is the CSR or CSR+ file.

The SmartCloud Cost Management Data Collector that you are using to collect your system data automates the data processing cycle. The collectors convert usage metering data created by your system into CSR or CSR+ files. Once the CSR or CSR+ files are created, the Processing Engine processes the data in the files and loads the resulting output files into the database using the components Acct.jar, Bill.jar, Load.jar, and the Sort subroutine.

Data processing frequency

The preferred method of processing is to run the full data processing cycle as the data becomes available. The data produced by the various operating systems (Mainframe, UNIX, Windows, etc.) and applications/databases (CICS®, DB2, Oracle, IIS, Exchange Server, etc.) are usually made available for processing on a daily basis.

Other feeds such as time accounting, help desk, line charges, equipment charges, and other shared services are usually produced on a monthly basis.

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

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

  Because there are fewer jobs, transactions, or records to review, the task of determining what caused the utilization spike is much simpler if caught on the day in which it occurred.
- If the Bill program 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 directory permissions for data processing

The administrator that executes processing using SmartCloud Cost Management Job Runner or Administration Console requires full access to files in the `processes` directory (that is, the ability to create, modify, delete, overwrite, etc.).

Therefore, the Linux user and group for the SmartCloud Cost Management administrator must have read, write, and execute permissions for the `processes` directory and all subdirectories. The Windows user account or group account for the administrator must have `Full Control` security permissions for the `processes` directory and all subdirectories.

SmartCloud Cost Management Processing Engine

SmartCloud Cost Management Processing Engine (`<SCCM_install_dir>/wlp/usr/servers/sccm/apps/sccm.ear/lib/aucProcessEngine.jar`) is composed of Java class files. Each engine component performs particular functions in the processing cycle.

The following table describes the components in the order that they appear in the processing cycle and the output files created by the objects as applicable.

Table 22. SmartCloud Cost Management Processing Engine Components

Component	Description	Output Files
`Acct`	**Note:** Acct was formerly CIMSAcct in previous releases of SmartCloud Cost Management. If you have upgraded to SmartCloud Cost Management 2.1.0.5 from a previous release and are using the CIMSAcct program it is recommended that you phase out the use of Acct and begin using the more powerful Integrator program **Note:** From 7.1.3 onwards usage of the ACCT component is not recommended for new deployments. Old jobfiles employing ACCT will continue to function however ACCT may be removed completely in a future release.. Performs account code conversion, shift determination, and date selection on the data contained in a CSR or CSR+ file.	• **Acct Output CSR+.** This file is input into Bill or can be reprocessed through Integrator or Acct.
`Bill`	Processes the sorted CSR or CSR+ file from Integrator or Acct and performs shift processing, CPU normalization, and include/exclude processing and creates files that contain the billing information used to generate invoices and reports. Also applies the rates to cost the data.	• **Resource.** This optional file is loaded into the database. • **Detail.** This file is loaded into the database. • **Summary.** This file is loaded into the database. • **Ident.** This file is loaded into the database.
`DBLoad`	Loads the Detail, Summary, and Ident files into the database.	

Table 22. SmartCloud Cost Management Processing Engine Components (continued)

Component	Description	Output Files
`ReBill`	The ReBill Stored Procedure will convert Money Value and Rate Value for a date range. It is really a simple version of CIMSBILL.	

Acct

The Acct program produces the Output CSR+ file, which contains records that are properly formatted for input into the Bill program.

Note: Acct was formerly CIMSAcct in previous releases of SmartCloud Cost Management. If you are have upgraded to SmartCloud Cost Management 2.1.0.5 from a previous release and are using the CIMSAcct program it is recommended that you phase out the use of Acct and begin using the more powerful Integrator program.

Acct performs the following functions:

- **Account code conversion.** Acct uses specified identifiers from the input file to build the account code and perform account code conversion if required.
- **Date selection.** Acct selects the input file records to be processed based on the specified date or date range.
- **Shift determination.** Acct can determine the shift to be used for processing in either of the following ways:
  - Use the shift code from the input file records. If a shift code is not included in the records, the default shift code is 1.
  - Recalculate the shift using the start date/time in the records.

Acct input

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

Table 23. Acct Input

File Type	File Name	Description
Input Files (Acct can process usage data from any of the following sources)		
CSR or CSR+ file	`CurrentCSR.txt` Or `yyyymmdd.txt`	These are the most commonly used files for processing data through SmartCloud Cost Management.
CIMSAcct Output CSR+ file	`AcctCSR.txt` (default) or User Defined	This file is the output from a previous run of Acct. This file provides an account code for each input file record. When used as input into Acct, this file is used primarily for further account code conversion.
Processing Files		

Table 23. Acct Input (continued)

File Type	File Name	Description
Control file	`AcctCntl.txt` (default) or User Defined	This file contains the Acct control statements. **Note:** The Acct control file is compatible with earlier versions and the use of the file is not recommended. It is recommended that you include control statements in the job file.
Account Code Conversion Table	`AcctTabl.txt` (default) or User Defined	This optional file contains the account code conversion table.

Acct output

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

Table 24. Acct Output

File Type	File Name	Description
Acct Output CSR+ file	`AcctCSR.txt` (default) or User Defined	This file provides an account code for each input file record.
Exception file	`Exception.txt` (default) or User Defined	This is an optional file in CSR or CSR+ format that contains all records that did not match during account code conversion. These files can be reprocessed through the processing cycle. To produce this file, you must have the EXCEPTION FILE PROCESSING ON control statement defined in the job file or defined the Acct control file, `AcctCntl.txt`, in the process definition directory that the job file points to.

Related concepts:

"Setting the account code conversion options (Acct program)" on page 167
This topic describes how the Acct program is used to implement account code conversion.

Related reference:

"Parameter element" on page 79
All Acct settings are documented under the Parameter element.

Bill

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

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

- If a resource has a corresponding entry with a monetary amount in the rate code table, the number of resource units are multiplied by this amount. The result appears in the Summary records. The exception is if the resource has a flat fee amount.

- If a resource has a corresponding entry with no or zero currency amount in the rate code table, a zero cost appears in the Summary records for that resource.
- If a resource *does not* have a corresponding entry in the rate code table, the data for that resource does not appear in the Summary records.
- If *none* of the resources in the Bill input file have a corresponding entry in the rate code table, the Summary file will contain no records. You must have at least one rate code from the input file in the rate code table to produce Summary records.

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

- Processes non-standard costs such as miscellaneous, recurring, and credit transactions.
- Performs CPU Normalization. .
- Performs include/exclude processing.

Bill input

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

Table 25. Bill Input

File Type	File Name	Description
Input Files (Bill can process usage data from any of the following sources)		
CSR or CSR+ file	`AcctCSR.txt` (default) or User Defined	This is the Output CSR or CSR+ file from Integrator or Acct
Processing Files		
Control file	`BillCntl.txt` (default) or User Defined	This file contains the Bill control statements. **Note:** The Bill control file is provided for backward compatibility and the use of the file is not recommended. It is recommended that you include control statements in the job file.

Bill output

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

Table 26. Bill Output

File Type	File Name	Description
Resource file	User Defined	This optional file is the same as the Detail file except that no CPU normalization or include/exclude processing has been performed and accounting dates are not set. This file is produced only when the Bill `resourceFile` attribute is provided in the job file and is loaded into the database only when the Load `loadType` attribute is set to `Resource` in the job file.
Detail file	`BillDetail.txt` (default) or User Defined	This file is loaded into the database for use in drill down reports. This file contains resource utilization and cost data. The Detail file differs from the Resource file in that the Detail file reflects any CPU normalization or include/exclude processing that was performed on the CSR or CSR+ file. The Detail file also includes accounting dates.
Summary file	`BillSummary.txt` (default) or User Defined	This file is loaded into the database for use in producing reports. This file contains both resource utilization and cost data.
Ident File	`Ident.txt` (default) or User Defined	This file contains all the identifiers that are contained in the input records (for example, user ID, jobname, department code, server name). These identifiers are used during account code conversion to create your target account code structure. This file is loaded into the database.

Related reference:

"Parameter element" on page 79
All Bill settings are documented under the Parameter element.

DBLoad

The DBLoad program loads the output files from the Bill program into the database.

By default, the DBLoad program loads the Detail, Ident, and Summary files into the SmartCloud Cost Management database. However, you can use the `loadType` attribute in the job file to specify all files (including the Resource file) or just specific files.

DBLoad input

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

Table 27. DBLoad Input

File Type	File Name	Description
Input Files		

Table 27. DBLoad Input (continued)

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

Related reference:

"Parameter element" on page 79
All DBLoad settings are documented under the Parameter element.

ReBill

The ReBill Stored Procedure will convert Money Value and Rate Value for a date range. It is really a simple version of CIMSBILL.

ReBill input

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

Table 28. ReBill Input

File Type	File Name	Description
Input Files (Bill can process usage data from any of the following sources)		
CSR or CSR+ file	`AcctCSR.txt` (default) or User Defined	This is the Output CSR or CSR+ file from Integrator or Acct
Processing Files		

Table 28. ReBill Input (continued)

File Type	File Name	Description
Control file	`BillCntl.txt` (default) or User Defined	This file contains the Bill control statements. **Note:** The Bill control file is compatible with earlier versions and the use of the file is not recommended. It is recommended that you include control statements in the job file.

ReBill output

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

Table 29. Bill Output

File Type	File Name	Description
Resource file	User Defined	This optional file is the same as the Detail file except that no CPU normalization or include/exclude processing has been performed and accounting dates are not set. This file is produced only when the Bill `resourceFile` attribute is provided in the job file and is loaded into the database only when the Load `loadType` attribute is set to `Resource` in the job file.
Detail file	`BillDetail.txt` (default) or User Defined	This file is loaded into the database for use in drill down reports. This file contains resource utilization data. The Detail file differs from the Resource file in that the Detail file reflects any CPU normalization or include/exclude processing that was performed on the CSR or CSR+ file. The Detail file also includes accounting dates.
Summary file	`BillSummary.txt` (default) or User Defined	This file is loaded into the database for use in producing reports. This file contains both resource utilization and cost data.
Ident File	`Ident.txt` (default) or User Defined	This file contains all the identifiers that are contained in the input records (for example, user ID, jobname, department code, server name). These identifiers are used during account code conversion to create your target account code structure. This file is loaded into the database.

Related reference:

"Parameter element" on page 79
All ReBill settings are documented under the Parameter element.

SmartCloud Cost Management Integrator

SmartCloud Cost Management Processing Engine (<*SCCM_install_dir*>/wlp/usr/servers/sccm/apps/sccm.ear/lib/aucIntegrator.jar)enables you to modify input data provided in a variety of formats (including CSR or CSR+ files). Integrator includes the functions provided by the Acct program (account code conversion, shift determination, and date selection) plus many other features such as adding an identifier or resource to a record, converting an identifier or resource to another value, or renaming identifiers and resources.

Note: The Acct program was formerly CIMSAcct in previous releases of SmartCloud Cost Management. If you have upgraded to SmartCloud Cost Management 2.1.0.5from a previous release and are using the CIMSAcct program it is recommended that you phase out the use of Acct and begin using the more powerful Integrator program.

Integrator Structure

Integrator processes the data in an input file according to stages that are defined in the job file XML. Each stage 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. You can add, remove, or activate, stages as needed.

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

Related concepts:

"Input element" on page 97
This topic describes all available options for the input element and how it is used in a job file.

"Stage elements" on page 100
This topic describes all stage elements and their options and they are used in a job file.

"Setting the account code conversion options (Acct program)" on page 167
This topic describes how the Integrator program is used to implement account code conversion.

Setting up and running job files

SmartCloud Cost Management Data Collectors read and convert usage metering data generated by applications (usually standard usage metering files such as log files) and produce a common output file, the CSR or CSR+ file, that is used by SmartCloud Cost Management. Once the CSR or CSR+ files are created, the data in the files is processed and the output is loaded into a database.

Related tasks:

Running sample proration
SmartCloud Cost Management includes a sample proration job file (<*SCCM_install_dir*>\samples\jobfiles\SampleProrate.xml. This job file processes a sample CSR file and produces an output CSR+ file with prorated values.

Creating job files

A job file is an XML file that defines the process of collecting data and loading it into a SmartCloud Cost Management database. Sample job files are provided by SmartCloud Cost Management Data Collector in the <*SCCM_install_dir*>/samples/jobfiles directory. You can use the XML in these sample files to create job files for your organization. The required and optional elements and attributes in a job file and their possible values are described in the References section of this Information Center.

Before you begin

Job files can be created using any XML or text editor. You can copy XML from another job file and paste it in the new job file. This option is useful as you can copy a job file that performs similar functions, and the file does not require as many edits. The file must be saved with an XML extension to the job file processing path.

What to do next

Job files contain XML parameters and parameter values. When a parameter value contains a greater than (>) or less than (<) character, you create or edit the job file by replacing the > and < characters with &gt and &lt, respectively.

For example:

```
<Parameter ParmVal2="> C:\netstatOut.txt"/>
```

Would be changed to:

```
<Parameter ParmVal2="&gt; C:\netstatOut.txt"/>
```

For example:

```
<Parameter ParmVal2="< C:\netstatOut.txt"/>
```

Would be changed to:

```
<Parameter ParmVal2="&lt; C:\netstatOut.txt"/>
```

You can run the updated job file from the command line.

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 SmartCloud Cost Management 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`.

The following table 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 email message. You can also use these attributes to send a separate email message for each individual job. These attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Table 30. Jobs Element Attributes

Attribute	Required or Optional	Description
`processFolder`	Optional	In most cases, you will not need to use this attribute. By default, the path to the `processes` directory set in the SmartCloud Cost Management ConfigOptions table is used. If you set this attribute, it will override the path to the`processes` directory that is set in the ConfigOptions table. **Example of the use of this attribute:** The `processFolder` attribute can be used to allow job processing to occur on a SmartCloud Cost Management application server that is not usually used for processing. For example, assume you have a single database server and process most of your feeds on a UNIX or Linux SmartCloud Cost Management application server. However, there are some feeds that you process on another SmartCloud Cost Management application server. You can configure the job file path in the database to point to the processes directory on the main server. On the other server, you can set the processFolder attribute in the job file to point to the processes path on that server. The result is that both SmartCloud Cost Management servers can use a single database, but process data on more than one server.
`smtpSendJobLog`	Optional	Specifies whether the job log should be sent via email. Valid values are: • `true`" (send via email) • `false`" (do not send) The default is "`false`".

Table 30. Jobs Element Attributes (continued)

Attribute	Required or Optional	Description
`smtpServer`	Optional	The name of the SMTP mail server that will be used to send the job log. The default is "`mail.ITUAMCustomerCompany.com`".
`smtpFrom`	Optional	The fully qualified email address of the email sender. The default is "`ITUAM@ITUAMCustomerCompany.com`".
`smtpTo`	Optional	The fully qualified email address of the email 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` • servername:userID:password:port: user@domain Example: `mail.xyzco.com:janes:global:25: jsmith@xyzco.com` If you want to use multiple addresses, separate them with a comma (,). 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`"
`smtpSubject`	Optional	The text that you want to appear in the email subject. The default subject is: `ITUAM job <job name> running on <server name> completed <successfully or with x warning(s)/with x error(s)>`
`smtpBody`	Optional	The text that you want to appear in the email body. The default body text is: `Attached are results from a JobRunner execution.`

Job Element

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.

The following table 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 31. Job Element Attributes

Attribute	Required or Optional	Description
`id`	Required	A text string name for the job. This value must be unique from other job ID values in the file. **Example** `id="Nightly"` In this example, the subfolder that contains log files for this job will also be named `Nightly`.
`description`	Optional	A text string description of the job (maximum of 255 characters). **Example** `description="Nightly collection and processing"`
`active`	Optional	Specifies whether the job should be run. Valid values are: • `true`" (run the job) • `false`" (do not run the job) The default is "`true`".
`dataSourceId`	Optional	The data source for the SmartCloud Cost Management database. **Example** `dataSourceId=ITUAMDev` If this parameter is not provided, the data source that is set as the default processing data source on Administration Console Data Source List Maintenance page is used. To use a data source other than the default, set this parameter to the appropriate data source ID
`joblogShowStepParameters`	Optional	Specifies whether parameters for the steps in a job 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 31. Job Element Attributes (continued)

Attribute	Required or Optional	Description
joblogShowStepOutput	Optional	Specifies whether output generated by the steps in a job 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".
processFolder	Optional	In most cases, you will not need to use this attribute. By default, the path to the processes directory set in the SmartCloud Cost Management ConfigOptions table is used. If you set this attribute, it will override the path to theprocesses directory that is set in the ConfigOptions table. **Example of the use of this attribute:** The processFolder attribute can be used to allow job processing to occur on a SmartCloud Cost Management application server that is not usually used for processing. For example, assume you have a single database server and process most of your feeds on a UNIX or Linux SmartCloud Cost Management application server. However, there are some feeds that you process on a Windows SmartCloud Cost Management application server. You can configure the job file path in the database to point to theprocesses directory on the UNIX server. On the Windows server, you can set the processFolder attribute in the job file to point to the processes path on the Windows server. The result is that both SmartCloud Cost Management servers can use a single database, but process data on more than one server platform.
processPriorityClass	Optional	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 Software Support 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.
joblogWriteToTextFile	Optional	Specifies whether the job log should be written to a text file. Valid values are: • true" (writes to a text file) • false" (does not write to a text file) The default is "true".
joblogWriteToXMLFile	Optional	Specifies whether the job log should be written to an XML file. Valid values are: • true" (writes to an XML file) • false" (does not write to an XML file) The default is "true".
smtpSendJobLog	Optional	Specifies whether the job log should be sent via email. Valid values are: • true" (send via email) • false" (do not send) The default is "false".

Table 31. Job Element Attributes (continued)

Attribute	Required or Optional	Description
smtpServer	Optional	The name of the SMTP mail server that will be used to send the job log. The default is "mail.ITUAMCustomerCompany.com".
smtpFrom	Optional	The fully qualified email address of the email sender. The default is "ITUAM@ITUAMCustomerCompany.com".
smtpTo	Optional	The fully qualified email address of the email receiver.
smtpSubject	Optional	The text that you want to appear in the email subject. The default subject is: ITUAM job <job name> running on <server name> completed <successfully or with x warning(s)/with x error(s)>
smtpBody	Optional	The text that you want to appear in the email body. The default body text is: Attached are results from a JobRunner execution.
stopOnProcessFailure	Optional	Specifies whether a job with multiple processes should stop if any of the processes fail. Valid values are: • true" (stop processing) • false" (continue processing) The default is "false". **Note:** If stopOnStepFailure is set to "false"at the Steps element level in a process, processing continues regardless of the value set for stopOnProcessFailure.

Process element

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 file system, for example).

The following table 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 32. Process Element Attributes

Attribute	Required or Optional	Description
`id`	Required	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. If the `buildProcessFolder` attribute is not included or is set to `"true"` (the default), 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="ABCSoftware"` In this example, the process definition folder created by Job Runner will be named `ABCSoftware`.
`description`	Optional	A text string description of the process (maximum of 255 characters). **Example** `description="Process for ABCSoftware"`
`buildProcessFolder`	Optional	Specifies whether Job Runner will create a process definition folder with the same name as the `id` attribute value in the `processes` folder. 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 Job Runner to run a script or program that does not require a process definition folder. 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"`.
`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 Software Support 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"`.

Steps Element

A `Steps` element is a container for one or more `Step` elements. The `Steps` element has one optional attribute.

Table 33. Steps Element Attribute

Attribute	Required or Optional	Description
`stopOnStepFailure`	Optional	Specifies whether processing should continue if any of the active steps in the process fail. Valid values are: • `true` (processing fails) If the `stopOnProcessFailure` attribute is also set to "`true`", the remaining processes in the job are not executed. If `stopOnProcessFailure` is set to "`false`", the remaining processes in the job are executed. • `false` (processing continues) In this situation, all remaining processes in the job are also executed regardless of the value set for `stopOnProcessFailure`. The default is "`true`".

Step Element

A `Step` element defines a step within a process.

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

The following table 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 34. Step Element Attributes

Attribute	Required or Optional	Description
`id`	Required	A text string name for the step. This value must be unique from other step ID values in the process. **Example** `id="Scan"` In this example, the step is executing the Scan program.
`description`	Optional	A text string description of the step (maximum of 255 characters). **Example** `description="Scan ABCSoftware"`

Table 34. Step Element Attributes (continued)

Attribute	Required or Optional	Description
`active`	Optional	Specifies whether the step should be run. Valid values are: • `true`" (run the step) • `false`" (do not run the step) The default is "`true`".
`type`	Required	The type of step that is being implemented: "`ConvertToCSR`" or "`Process`". `ConvertToCSR` specifies that the step performs data collection and conversion and creates a CSR file. `Process` specifies that the step executes a program such as Scan, Acct, Bill, and so forth.
`programName`	Required	The name of the program that will be run by the step. It the program name is any of the following, the `programType` attribute must be `java`: • Integrator • SendMail • Acct • Bill • Sort • DBLoad • DBPurge • JobFileConversion • Rpd • Scan • Cleanup • FileTransfer • WaitFile

Table 34. Step Element Attributes (continued)

Attribute	Required or Optional	Description
		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, Job Runner searches the `collectors`, `bin`, and `lib` directories for the program in the order presented. If the `type` attribute is `Process`, this value is the name of a SmartCloud Cost Management program (for example, `"Scan"`, `"Acct"`,`"Bill"`, `"DBLoad"`, and so forth). Examples: programName="WinDisk.exe" `programName="Cleanup"`
`processPriorityClass`	Optional	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 Software Support 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.
`programType`	Optional	The type of program specified by the `programName` attribute: • `"console"`-Console Application • `"com"`-COM Component (deprecated, compatible with earlier versions only) • `"net"`-.Net Component • `"java"`-Java application

Table 34. Step Element Attributes (continued)

Attribute	Required or Optional	Description
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".
joblogShowStepOutput	Optional	Specifies whether output generated by the step 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".

Parameters Element

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

Parameter element

A Parameter element defines a parameter to a step.

The valid attributes for collection 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.

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 Acct and Bill, 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 Acct and the proration table used by Integrator. You can place these files in a central location so that they can be used by multiple processes. In this case, you must provide the path.
- Attributes include macro capability so that the following predefined 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.
  - **%CollectorLogs%**. Specifies the collector log files folder as defined as in the CIMSConfigOptions table.
  - **%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.
  - **%<*Date Keyword*>_Start%**. For files that contain a date in the file name, specifies that files with dates matching the first day of the <*Date Keyword*>

parameter value are used. For example, if the *<Date Keyword>* 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.

- **%*<Date Keyword>*_End%**. For files that contain a date in the file name, specifies that files with dates matching the last day of the *<Date Keyword>* parameter value are used. For example, if the *<Date Keyword>* 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 in the file name, specifies that files with dates 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.
- **%LogDate_End%.** For files that contain a date in the file name, specifies that files with dates 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.

The valid attributes for process step parameters (type=Process) are listed in the following sections. The attributes are broken down as follows

- Parameter attributes that are specific to a program (Scan, Acct, Bill, etc.).
- Parameter attributes that are specific to a program type (wsf, com, net, console, etc.).

Acct specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the acct program.

Attributes

Table 35. Acct specific parameter attributes

Attribute	Required or Optional	Description
accCodeConvTable	Optional	The name of account code conversion table used by Acct. Include a path if the table is in a location other than the collector's process definition directory. **Examples:** `accCodeConvTable="MyAcctTbl.txt"` `accCodeConvTable="E:\Processes\Account\ MyAcctTbl.txt` The default is "AcctTabl.txt".

Table 35. Acct specific parameter attributes (continued)

Attribute	Required or Optional	Description
controlCard	Optional	A valid Acct control statement or statements. All Acct control statements are stored in the Acct control file. **Note:** If you have an existing Acct control file in the process definition directory, the statements that you define as controlCard parameters will overwrite all statements currently in the file. To define multiple control statements, use a separate parameter for each statement. **Example** <Parameter controlCard="TEST A"/ <Parameter controlCard="VERIFY DATA ON"/>
controlFile	Optional	The name of the control file used by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** controlFile="MyAcctCntl.txt" The default is "AcctCntl.txt".
exceptionFile	Optional	The name of the exception file produced by Acct. This file must be in the collector's process definition directory-do not include a path. The file name should contain the log date so that it is not overwritten when Acct is run again. **Example** exceptionFile= "Exception_%LogDate_End%.txt" The default is "Exception.txt".
inputFile		The name of CSR or CSR+ file to be processed by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** inputFile="MyCSR.txt" The default is "CurrentCSR.txt".
outputFile		The name of the output CSR or CSR+ file produced by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** outputFile="CSR.txt" The default is "AcctCSR.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example account code conversion traces) • "false" (Less detailed information is provided) The default is "false".

Bill specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the bill program.

Attributes

Table 36. Bill specific parameter attributes

Attribute	Required or Optional	Description
`controlCard`	Optional	A valid Bill control statement or statements. All Bill control statements are stored in the Bill control file. **Note:** If you have an existing Bill control file in the process definition directory, the statements that you define as `controlCard` parameters will overwrite all statements currently in the file. To define multiple control statements, use a separate parameter for each statement. **Example** `<Parameter controlCard="CLIENT SEARCH ON"/>` `<Parameter controlCard="DEFINE J1 1 1"/>`
`controlFile`	Optional	The name of the control file used by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `controlFile="MyBillCntl.txt"` The default is `"BillCntl.txt"`.
`detailFile`	Optional	The name of the Detail file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `detailFile=` `"MyDetail.txt"` The default is `"BillDetail.txt"`.
`dateSelection`	Optional	Defines a date range for records to be processed by Bill. Valid values are a from and to date range in yyyymmdd format or a date keyword. **Examples** `dateSelection="2008117 2008118"` In this example, Bill will process records with an accounting end dates of January 17 and 18, 2007. `dateSelection="PREDAY"` In this example, Bill will process records with an accounting end date one day prior to the date Job Runner is run.
`defaultRateTable`	Optional	Defines the default Rate Table to be used when matching a Resource Entry to a Rate. The `'Standard'` Rate table is used if this option is not specified.
`identFile`	Optional	The name of the Ident file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `identFile="MyIdent.txt"` The default is `"Ident.txt"`.

Table 36. Bill specific parameter attributes (continued)

Attribute	Required or Optional	Description
inputFile		The name of the CSR or CSR+ file to be processed by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** inputFile="CSR.txt" The default is "AcctCSR.txt".
keepZeroValueResources	Optional	This parameter when set to true, enables resources with zero values to be written to or read from CSR files and in billing output. Resources with zero values are normally discarded. The default is "false".
multTableFile	Optional	The name of the proration table used by Prorate. Include a path if the table is in a location other than the collector's process definition directory. **Examples** multTableFile="MyMultTable.txt" multTableFile="E:\Processes\Prorate\MyMultTable.txt"
rateSelection	Optional	Defines the algorithm used to determine what Historical Rate over an Accounting period should be used when matching a Resource Entry to a Rate. Valid options are NONE, FIRST, and LAST. The default value is NONE. • NONE: If only one rate is effective over the Accounting period, that rate is used. If more than one rate is effective over the Accounting period, no Rate is matched. • FIRST: The first rate effective over the Accounting period is used. • LAST: The last rate effective over the Accounting period is used.
reportDate	Optional	Defines the dates that are used as the accounting start and end dates in the Summary records created by Bill. Valid values are a date in yyyymmdd format or a 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 "CURMON", which sets the accounting start date to the first of the month and the end date to the last day of the month.
resourceFile		The name of the Resource file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** resourceFile="MyResource.txt" There is no default. This file is not produced if this attribute is not provided.
summaryFile	Optional	The name of the Summary file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** summaryFile="MySummary.txt" The default is "BillSummary.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example accounting date calculation tracing and client search tracing) • "false" (Less detailed information is provided) The default is "false".

Cleanup specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the Cleanup program.

Attributes

Table 37. Cleanup specific parameter attributes

Attribute	Required or Optional	Description
`dateToRetainFiles`	Optional	A date by which all yyyymmdd files that were created prior to this date will be deleted. You can use a date keyword or the date in yyyymmdd format. **Example** `dateToRetainFiles="PREMON"` This example specifies that all files that were created prior to the previous month will be deleted.
`daysToRetainFiles`		The number of days that you want to keep the yyyymmdd files after their creation date. **Example** `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.
`cleanSubfolders`	Optional	Specifies whether the files that are contained in subdirectories are deleted. Valid values are: • `"true"` (the files are deleted) • `"false"` (the files are not deleted) The default is `"false"`.
`folder`	Optional	By default, the Cleanup program deletes files with file names containing the date in yyyymmdd format from the collector's process definition directory. If you want to delete files from another directory, use this attribute to specify the path and directory name. **Example** `folder="\\Server1\LogFiles`

Console parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the CONSOLE program type.

Attributes

Table 38. Console parameter attributes

Attribute	Required or Optional	Description
`scanFile`	Optional	This attribute is applicable only if the Smart Scan feature is enabled . 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. 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` 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.

Console specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the CONSOLE program type.

Attributes

Table 39. Console specific parameter attributes

Attribute	Required or Optional	Description
`useCommandProcessor`	Optional	Specifies whether the `Cmd.exe` program should be used to execute a console program. If the `Cmd.exe` program is not used, then the console program is called using APIs. Valid values are: • `"true"` (the `Cmd.exe` program is used) • `"false"` (the `Cmd.exe` program is not used) The default is `"true"`.

Table 39. Console specific parameter attributes (continued)

Attribute	Required or Optional	Description
`useStandardParameters`		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: • `LogDate` • `RetentionFlag` • `Feed` • `OutputFolder` These parameters are passed before any other parameters defined for the step. Valid values are: • `"true"` (the standard parameters are passed) • `"false"` (the standard parameters are not passed) If the step type is `Process`, the default value is `"false"`. If the step type is `ConvertToCSR`, the default is `"true"`.
`XMLFileName`, `CollectorName`, and `CollectorInstance`	Optional	These attributes are used by the Windows Disk and Windows Event Log collectors. They specify the name of the XML file used by the collector; the name of the collector; and the collector instance, respectively.

DBLoad specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the DBLoad program.

Attributes

Table 40. DBLoad specific parameter attributes

Attribute	Required or Optional	Description
`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"`.
`bypassDetailDuplicateCheck`	Optional	Allows the user to skip the Detail Duplicate Check. although the Summary Duplicate Check is still performed. It is recommended that this duplicate check is performed. However, you may want to skip this check to avoid unnecessary overhead or to increase the speed of load. Valid values are: • `"true"` (bypass detail duplicate check) • `"false"` (perform detail duplicate check) The default is `"false"`.

Table 40. DBLoad specific parameter attributes (continued)

Attribute	Required or Optional	Description
bypassDuplicateCheck	Optional	Allows the user to skip the Detail and Summary Duplicate Check. It is recommended that these duplicate checks are performed. However, you may want to skip these checks to avoid unnecessary overhead or to increase the speed of load. Valid values are: • "true" (bypass detail and summary duplicate checks) • "false" (perform detail and summary duplicate checks) The default is "false".
detailFile	Optional	The name of the Detail file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. detailFile= "MyDetail.txt" The default is "BillDetail.txt".
identFile	Optional	The name of the Ident file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** identFile="MyIdent.txt" The default is "Ident.txt".
loadType		By default, the DBLoad program loads the Summary, Detail, Ident, and Resource (optional) files into the database. If you want to load a specific file rather than all files, the valid values are: • Summary • Detail • Resource • Ident • DetailIdent (loads Detail and Ident files) • All (loads Summary, Detail, and Ident file types)
resourceFile		The name of the Resource file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** resourceFile="MyResource.txt" There is no default. This file is not produced if this attribute is not provided.
summaryFile	Optional	The name of the Summary file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** summaryFile="MySummary.txt" The default is "BillSummary.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example accounting date calculation tracing and client search tracing) • "false" (Less detailed information is provided) The default is "false".

Table 40. DBLoad specific parameter attributes (continued)

Attribute	Required or Optional	Description
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".

DBPurge specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the DBPurge program.

Attributes

Note: For an example of the use of DBPure in a job file, see the SampleDBPurge.xml file in <*SCCM_install_dir*>\samples\jobfiles\.

Table 41. DBPurge specific parameter attributes

Attribute	Required or Optional	Description
MonthsToKeep	Optional	Specifies the age of the loads (in months) that you want to delete from the tables.
PurgeSummary, PurgeBillDetail, PurgeIdent, and PurgeAcctDetail.	Required	Specify whether the CIMSSummary, CIMSDetail, CIMSDetailIdent, or CIMSResourceUtilization tables are purged. Valid values are: • "true" (The table is purged) • "false" (The table is not purged)
StartDate and EndDate **Note:** MonthsToKeep parameter and the StartDate and StopDate parameters are mutually exclusive. If all parameters are specified, StartDate and StopDate parameters are ignored.	Optional	Specifies the date range for the loads that you want to delete from the tables. Any loads that have accounting start and end dates in this range are deleted. Valid values are: • preday (previous day) • rndate (current day) • premon (previous month) • curmon (current month) • date in yyyymmdd format If you use the premon or curmon keyword for the start date or end date, the first day of the month is used for the start date and the last day of the month is used for the end date.

FileTransfer specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the FileTransfer program.

Parameters

Table 42. FileTransfer specific parameter attributes

Attribute	Required or Optional	Description
`continueOnError`		For a multi-file transfer, specifies whether subsequent file transfers continue if a transfer fails. Valid values are: • `"true"` (file transfer continues) • `"false"` (file transfer does not continue) The default is `"false"`.
`pollingInterval`		The number of seconds to check for file availability (maximum of 10,080 [one week]). **Example** `pollingInterval="60"` This example specifies a polling interval of 60 seconds. The default is 5 seconds.
`type`	Required	The type of file transfer. Valid values are: • `"ftp"` (File Transfer Protocol [FTP] transfer) • “file” (Local transfer) • “ssh” (Secure Shell transfer)
`UseSFTP`	Optional	It the `type` attribute value is `"ssh"`, this attribute specifies whether the SFTP or SCP protocol will be used for transferring files. Valid values are: • `"true"` (the SFTP protocol is used) • `"false"` (the SCP protocol is used) The default is `"false"`. A value of `"true"` is used with certain SSH servers (such as Tectia SSH Server 5.x) to allow file transfers to complete successfully.

The following attributes `from`, `to`, `action`, and `overwrite` are attributes of a single `Parameter` element. If you are transferring multiple files, include a `Parameter` element with these attributes for each file. For an example of these attributes in a job file, see the `SampleNightly.xml` file.

Table 43. FileTransfer parameters

Attribute	Required or Optional	Description
`action`	Required	Specifies the file activity. Valid values are: • `"Copy"` (copies the file from the `from` location to the `to` location) • `"Delete"` (deletes the file from the `from` location) • `"Move"` (copies the file from the `from` location to the `to` location and then deletes the file from the `from` location) The default is `Copy`.

Table 43. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
from and to	Required	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: • type="ftp" The "ftp://" file transfer protocol can be used for either the from or the to location. However, because the job file must be run from the SmartCloud Cost Management application server system, the file transfer samples show the "ftp://" file transfer protocol being used only in the from parameter. Note the following when using the from and to attributes with type="ftp": – Values must be specified for serverName, userId, and userPassword parameters. – The transferType attribute is optional. Valid values are binary (default) or ascii. – The from and to parameters can include SmartCloud Cost Management macros (for example, %LogDate_End%). – The from location can include wildcards in the file name. If the from parameter includes file name wildcards, the to parameter must specify a directory name (no file names). For an example of the use of the from and to parameters for an FTP transfer, refer to the sample job file SampleFileTransferFTP_withPW.xml.
		• type="file" The "file://" file transfer protocol is used for both the from and to parameters. Note the following when using the from and to attributes with type="file" – The serverName, userId, and userPassword parameters are not required. – The from location can be a local path, a Windows UNC path, a Windows mapped drive, or a UNIX mounted drive path. – The from location must be a path that is local to the from location. On Windows, this can include a local hard disk drive, a Windows UNC path, or a Windows mapped drive. On UNIX, this can include a local directory or a UNIX mounted drive path – The from and to parameters can include SmartCloud Cost Management macros (for example, %LogDate_End%). – The from location can include wildcards in the file name. If the from parameter includes file name wildcards, the to parameter must specify a directory name (no file names). For an example of the use of the from and to parameters for a Windows transfer, refer to the sample job file SampleFileTransferLocal_noPW.xml.

Table 43. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
		• `type="ssh"` The `"ssh://"` file transfer protocol can be used for either the `from` or the `to` location. However, because the job file must be run from the SmartCloud Cost Management application server system, the file transfer samples show the `"ssh://"` file transfer protocol being used only in the `from` parameter. Note the following when using the `from` and `to` attributes with `type="ssh"` – Values must be specified for `serverName`, `userId`, and `userPassword` parameters. – If you are using a keyfile to authenticate with the the ssh daemon (sshd on UNIX or Linux), refer the information in the "Sample SSH file transfer job file" section of *Administering data processing*. - The user ID on the remote system must be configured to trust the SmartCloud Cost Management application server user ID. This means that if the keyfile path is specified, the public key for the SmartCloud Cost Management application server user ID must be in the authorized keys file of the remote user. – The `from` and `to` parameters must use the UNIX-style naming convention, which uses forward slashes (for example, `ssh:///anInstalledCollectorDir/CollectorData.txt`), regardless of whether you are transferring files from a UNIX or Linux system or a Microsoft Windows system. – You can use the ssh file transfer protocol on UNIX or Linux systems. However, before you run a job file that uses the ssh protocol, verify that you can successfully issue ssh commands from the SmartCloud Cost Management application server to the remote ssh system. – The `from` and `to` parameters can include SmartCloud Cost Management macros (for example, `%LogDate_End%`). – The `from` location can include wildcards in the file name. If the `from` parameter includes file name wildcards, the `to` parameter must specify a directory name (no file names).
`overwrite`	Optional	Specifies whether the destination file is overwritten. Valid values are: • `"true"` (the file is overwritten) • `"false"` (the file is not overwritten) The default is `"false"`.

The following attributes are for FTP transfer only.

Table 44. FileTransfer parameters

Attribute	Required or Optional	Description
`connectionType`	Optional	Describes how the connection address is resolved. This is an advanced configuration option that should be used only after consulting IBM Software Support. Valid values are: • `"PRECONFIG"` (retrieves the proxy or direct configuration from the registry) • `"DIRECT"` (resolves all host names locally) • `"NOAUTOPROXY"` (retrieves the proxy or direct configuration from the registry and prevents the use of a startup Microsoft JScript or Internet Setup (INS) file) • `"PROXY"` (passes requests to the proxy unless a proxy bypass list is supplied and the name to be resolved bypasses the proxy) The default is `"PRECONFIG"`.

Table 44. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
passive	Optional	Forces the use of FTP passive semantics. In passive mode FTP, the client initiates both connections to the server. This solves the problem of firewalls filtering the incoming data port connection to the FTP client from the FTP server. This is an advanced configuration option that should be used only after consulting IBM Software Support.
proxyServerBypass	Optional	This is a pointer to a null-terminated string that specifies an optional comma-separated list of host names, IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. This options is used only when connectionType="PROXY". This is an advanced configuration option that should be used only after consulting IBM Software Support.
proxyServer	Optional	If connectionType="PROXY", the name of the proxy server(s) to use. This is an advanced configuration option that should be used only after consulting IBM Software Support.
serverName	Required	A valid FTP IP address or server name. Example: serverName="ftp.xyzco.com"
transferType	Optional	The type of file transfer. Valid values are: • "binary" • "ascii" The default is "binary".
IsZOS	Optional	Indicates whether the FTP connection is to a Z/OS file system or not. Valid values are: • "true" • "false" The default value is "false"
userId	Optional	The user ID used to log on to the FTP server.
userPassword	Optional	The user password used to log on to the FTP server.

Rebill specific parameter attributes:

This topic outlines all the possible settings that can be entered using the parameter element in the Rebill program.

Attributes

Table 45. Rebill specific parameter attributes

Attribute	Required or Optional	Description
DataSourceName	Optional	This allows you to set the data source to use for the ReBill operation. Default is the processing data source.
EndDate	Required	The end date (YYYYMMDD) of the range in the CIMSSummary table that will be converted. Can also pass keywords such as CURDAY, PREDAY, RNDATE, PREMON or CURMON. If PREMON or CURMON are passed, it will use a range for StartDate/EndDate.
RateCode	Required	The rate code that should be converted. Default is all.

Table 45. Rebill specific parameter attributes (continued)

Attribute	Required or Optional	Description
`RateTable`	Optional	The rate table for the rate codes that should be converted. For the standard rate table this would be: `Parameter RateTable="STANDARD"` .
`StartDate`	Required	The start date (YYYYMMDD) of the range in the CIMSSummary table that will be converted. Can also pass keywords such as `CURDAY`, `PREDAY`, `RNDATE`, `PREMON` or `CURMON`. If `PREMON` or `CURMON` are passed, it will use a range for `StartDate/EndDate`.

Scan specific parameter attributes:

This topic outlines all the possible attributes that can be set using the parameter element in the scan program.

Attributes

Table 46. Scan specific parameter attributes

Header	Header	Header
`allowEmptyFiles`	Optional	Specifies whether a warning or error occurs when feed subdirectories contain a zero-length file that matches the log date value. Valid values are: • `"true"` (a warning occurs, processing continues) • `"false"` (an error occurs, processing fails) The default is `"false"`.
`allowMissingFiles`	Optional	Specifies whether a warning or error occurs when feed subdirectories do not contain a file that matches the log date value. Valid values are: • `"true"` (a warning occurs, processing continues) • `"false"` (an error occurs, processing fails) The default is `"false"`.
`excludeFile`	Optional	The name of a file to be excluded from the Scan process. The file can be in any feed subdirectory in the collector's process definition folder. The file name can include wildcard characters but not a path. **Example** `excludeFile="MyCSR*"` In this example, all files that begin with `MyCSR` are not scanned.
`excludeFolder`	Optional	The name of a feed subfolder to be excluded from the Scan process. The subdirectory name can include wildcard characters but not a path. The feed directory must be a top-level directory within the process definition folder. **Example** `excludeFolder="Server1"` In this example, the feed subdirectory `Server1` is not scanned.

Table 46. Scan specific parameter attributes (continued)

Header	Header	Header
includeFile	Optional	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 subdirectory in collector's process definition directory. **Example** `includeFile="MyCSR.txt"` In this example, files in the feed subdirectorys that are named MyCSR are scanned.
retainFileDate	Optional	Specifies whether the date is retained in the final CSR file (i.e., yyyymmdd.txt rather than CurrentCSR.txt). Valid values are: • "true" (the file name is yyyymmdd.txt) • "false" (the file name is CurrentCSR.txt) The default is "false".
useStepFiles	Optional	Specifies whether the Smart Scan feature is enabled. 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 subdirectories (for example, CIMSWinProcess/Server1/20080624.txt). If you want to override the default name, use the parameter attribute scanFile in the collection step.

WaitFile specific parameter attributes:

This topic outlines all the possible attributes that can be entered using the parameter element in the WaitFile program.

Attributes

Table 47. WaitFile specific parameter attributes

Attribute	Required or Optional	Description
pollingInterval		The number of seconds to check for file availability (maximum of 10,080 [one week]). **Example** `pollingInterval="60"` This example specifies a polling interval of 60 seconds. The default is 5 seconds.
timeout	Optional	The number of seconds that Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails. **Example** `timeout="18000"` This example specifies a timeout of 5 hours. The default is to wait indefinitely.

Table 47. WaitFile specific parameter attributes (continued)

Attribute	Required or Optional	Description
timeoutDateTime	Optional	A date and time up to which 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. **Example** `timeoutDateTime="%rndate% 23:59:59"` This example specifies a timeout of 23:59:59 on the day Job Runner is run. The default is to wait indefinitely.
filename	Required	The name of the file to wait for. If a path is not specified, the path to the process definition directory 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="BillSummary_` `%LogDate_End%.txt"` In this example, Job Runner will wait for Summary files that contain the same end date as the %LogDate_End% value.

Defaults Element

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

Default Element

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.

There are two types of attributes that you can use in a Default element: pre-defined and user defined as shown in the following table.

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 48. Default Element Attributes

Attribute	Description
Pre-defined attributes. These are the attributes that are pre-defined for SmartCloud Cost Management.	
LogDate	The log date specifies the date for the data that you want to collect. You should enter the log date in the job file only if you are running a snapshot collector or the Transactions collector.
RetentionFlag	This attribute is for future use. Valid values are KEEP or DISCARD.

Table 48. Default Element Attributes (continued)

Attribute	Description
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.

Default Example

This job file example contains two Default elements.

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

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

```
<?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">
   <Defaults>
   <Default programName="CIMSACCT"
   accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\
   AcctTabl-Win.txt"/>
   </Defaults>
   <Process   id="DBSpace"
   description="Process for DBSpace Collection"
    active="true">
   <Defaults>
   <Default LogDate="RNDATE"/>
   </Defaults>
   <Steps>
   :
   :
```

Integrator job file structure

The Integrator provides a flexible means of modifying the input data that you collect from usage metering files. Integrator processes the data in an input file according to stages that are defined in the job file XML.

Each stage 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. You can add, remove, or activate, stages as needed.

Note: In addition to usage metering files, you can also use integrator to collect data from a variety of other files, including CSR and CSR+ files.

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

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 identifies the type of file to be processed.

Example

In the following example, the input file is a CSR file.

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

Where:

- The `Input name` value can be:
  - **`"CSRInput"`**

    This value is used to parse a CSR or CSR+ type of file. An example of this value is provided in most sample collector job files.
  - **`"AIXAAInput"`**

This value is used to parse data in an Advanced Accounting log file. For an example of the use of this value, see the SampleAIXAA.xml job file.

- **"ApacheCommonLogFormat"**

  This value is used to parse data in an Apache log where the records are in the Apache Common Log Format (CLF). For an example of the use of this value, see the SampleApache.xml job file.

- **"DBSpaceInput"**

  This value is used to parse data gathered directly from a SQL Server or Sybase database or databases. For an example of the use of this value, see the SampleDBSpace.xml job file.

- **"MSExchange2003"**

  This value is used to parse data in a Microsoft Exchange Server 2000 or 2003 log file that records the activities on the server. For an example of the use of this value, see the SampleExchangeLog.xml job file.

- **"NCSAInput"**

  This value is used to parse data in a WebSphere HTTP Server access log. For an example of the use of this value, see the SampleWebSphereHTTP.xml job file.

- **"NotesDatabaseSizeInput"**

  This value is used to parse data in a Notes® Catalog database (catalog.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

- **"NotesEmailInput"**

  This value is used to parse data in a Notes Log Analysis database (loga4.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

- **"NotesUsageInput"**

  This value is used to parse data in a Notes Log database (catalog.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

- **"SAPInput"**

  This value is used to parse data in an SAP log file.

- **"SAPST03NInput"**

  This value is used to parse a data in an SAP Transaction Profile report.

- **"VmstatInput"**

  This value is used to parse data in a vmstat log file. For an example of the use of this value, see the SampleVmstat.xml job file.

- **"SarInput"**

  This value is used to parse data in a sar log file. For an example of the use of this value, see the SampleSar.xml job file.

- **"W3CWinLog"**

  This value is used to parse data in a Microsoft Internet Information Services (IIS) log file. For an example of the use of this value, see the SampleMSIIS.xml job file.

- **"HPVMSarInput"**

  This value is used to parse data in a HP VM sar log file. For an example of the use of this value, see the SampleHPVMSar.xml job file.

- **"KVMInput"**

  This value is used to parse data in a KVM log file. For an example of the use of this value, see the SampleKVM.xml job file.

- **"CollectorInput"**

This value is followed by one of the following `Collector name` attributes:

- **DATABASE, DELIMITED, FIXEDFIELD, or REGEX**

  These value are used to

  - Collect data from a database.
  - Parse data in a file that contains delimited record fields.
  - Parse data in a file that contains with fixed record fields.
  - Parse data in a file that contains delimited or fixed record fields using a regular expression.

  For an example of the use of the `DELIMITED` value, see the `SampleUniversal.xml` job file.

- **EXCHANGE2007**

  This value is used to parse data in a Microsoft Exchange Server 2007 message log file. For an example of the use of this value, see the `SampleExchange2007Log.xml` job file.

- **HMC**

  This value is used to collect data from HMC Server. For an example of the use of this value, see the `SampleHMC.xml` job file.

- **HMCINPUT**

  This value is used to parse data from HMC Server. For an example of the use of this value, see the `SampleHMC.xml` job file.

- **NFC**

  This value is used to parse data from NFC. For an example of the use of this value, see the `SampleNetworkFlow.xml` job file.

- **REGEX**

  This value is used to parse data in a record regular expression which is used to parse the record

- **TDS**

  This value is used to collect data from a TDSz DB2 database. For an example of the use of this value, see the `SampleTDSz.xml` job file.

- **TPC**

  This value is used to parse data in a TPC log file. For an example of the use of this value, see the `SampleTPC.xml` job file.

- **TPCVIEW**

  This value is used to collect data from TPC DB Data View. For an example of the use of this value, see the `SampleTPC_DiskLevel.xml` and `SampleTPC_StorageSubsystemLevel.xml` job files.

- **TRANSACTION**

  This value is used to collect data from the Transaction table in the SmartCloud Cost Management database. For an example of the use of this value, see the `SampleTransaction.xml` job file.

- **TSM**

  This value is used to collect data from a TSM DB2 database. For an example of the use of this value, see the `SampleTSM.xml` job file.

- **VMWARE**

  This value is used to collect data from VMware VirtualCenter or VMware Server Web service. For an example of the use of this value, see the `SampleVMWare.xml` job file.

- **WEBSPHEREXDFINEGRAIN**

This value is used to parse data in a WebSphere Extended Deployment (XD) Fine-Grained Power Consumption log file `FineGrainedPowerConsumptionStatsCache.log`. For an example of the use of this value, see the `SampleWebSphereXDFineGrain.xml` job file.

- **WEBSPHEREXDSERVER**

  This value is used to parse data in a WebSphere Extended Deployment (XD) Server Power Consumption log file `ServerPowerConsumptionStatsCache.log`. For an example of the use of this value, see the `SampleWebSphereXDFineGrain.xml` job file.

- The `File 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:

This section describes the Stage elements.

Aggregator:

The `Aggregator` stage aggregates a file based on the identifiers and resources specified. Any resources and identifiers not specified are dropped from the record. The Aggregator stage is memory dependent; the amount of memory affects the amount of time it takes to perform aggregation.

Settings

The Aggregator stage accepts the following input elements.

Attributes

The following attributes can be set in the `<Aggregator>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of a resource to aggregate. This setting can be used multiple times to name multiple resources. If a resource is not named it will not be aggregated and will not be included in the output file.
- `aggregationfunction="sum | min | max | avg | none"`: The aggregation type. The default aggregation operator for resources is to sum all resource values. Additional aggregation functions are also available. The available aggregation functions are:
  - `SUM`: (the default) sums all resources producing a total
  - `MIN`: finds the minimum value of a resource within an aggregate
  - `MAX`: finds the maximum value of a resource within an aggregate
  - `AVG`: produces the average of a series of resource values within an aggregate
  - `NONE`: does not apply any aggregation function (useful for values such as the number of CPUs)

This setting is used as follows:

```
<Resource name="RESSUM"  aggregationfunction = "avg"/>
```

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: This setting can be used multiple times to name multiple identifiers. If an identifier is not named it will not be aggregated and will not be included in the output file. The order in which records are placed in the output file is set by the order in which the identifiers are defined. Precedence is established in sequential order from the first identifier defined to the last. If a defined identifier appears in a record with a blank value, it will be included in the aggregated record with a blank value.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `defaultAggregation="true | false"`: This setting specifies whether the fields in the record header (start date, end date, account code, etc.) are used for aggregation. The default setting is true.
- `impliedZeroForMissingResources="true | false"`: There may be some scenarios that require a missing resource to be accounted for. This setting that allows you to add this behavior. For more information see "Implied zero for missing resources" on page 103.
- `includeNumRcdsResource="true | false"`: Setting this to true will cause the number of records within an aggregate to be counted using the includeNumRcdsResource option. A resource called Num_Rcds is generated listing the number. The default setting is false.
- `aggregationIntervalHours="num_hours"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of hours in that interval.
- `aggregationIntervalMinutes="num_minutes"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of minutes in that interval.
- `aggregationIntervalSeconds="num_seconds"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of seconds in that interval.
- `aggregationIntervalValidation="true | false"`: Setting this to true will catch end time - start time values which exceed the specified aggregation interval. For example, if the aggregation interval is 5 minutes, and the record has data from a 10 minute interval, a warning is logged in the trace file.

Example

This is a basic example using the default aggregation operator, SUM. Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `Aggregator` stage appears as follows:

```
<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:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXEMRCV,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXEMRCV,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,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. In the example, the identifier `User` is defined first. As a result, the output records are ordered based on user.

Aggregation by time interval:

In addition to aggregation functions, the aggregator can aggregate resource values into time intervals.

In addition to aggregation functions, the aggregator can aggregate resource values into time intervals. By default, time values in are ignored and only the identifiers specified in the `Identifiers` element are used to aggregate a series of records. However, if an aggregation interval value is specified, the aggregates will be grouped based on a time interval starting at midnight, based on the end time value. For example, if an aggregation interval of 5 minutes is specified, aggregator will create aggregates for resource records falling within the following time periods:

00:00:00 through 00:04:59

00:05:00 through 00:09:59

00:10:00 through 00:14:59

:

23:55:00 through 23:59:59

The end-time value of the incoming CSR record is used to determine which aggregate time slot the CSR record resource values are aggregated into it. The `aggregationIntervalValidation` parameter can be used to catch end-time to start-time values which exceed the specified aggregation interval. For example, if the aggregation interval is five minutes, and the record has data from a 10 minute interval, a warning is logged in the trace file.

The aggregation interval can be specified in seconds, minutes, or hours using the following parameters:

- aggregationIntervalSeconds
- aggregationIntervalMinutes
- aggregationIntervalHours

Implied zero for missing resources:

There may be some scenarios that require a missing resource to be accounted for. The setting that allows you to add this behavior is impliedZeroForMissingResources.

In some scenarios, one or more resources may not appear in all records. There is a default assumption that a resource missing from an input record means the resource is to be ignored for that record. However, there may be some scenarios that require that the missing resource to be accounted for. The option to control this behavior is impliedZeroForMissingResources, and is defined as:

```
<Parameter impliedZeroForMissingResources="false"/>
```

where "false", the default setting, assumes the metric is to be ignored, and "true" assumes the metric is to be accounted for.

If the option is set to true and a resource is missing from an input record, the resource is added with a value of zero.

This option primarily affects the calculation of the AVG function. For example, assume the following two input records having the resources number of CPUs (NUM_CPU) and memory used, (MEM_USED) and that the AVG function will be applied to both metrics:

```
Record #1: User1, NUM_CPU=2,  MEM_USED=4
Record #2: User1, MEM_USED=4
```

Notice in Record #2 the NUM_CPU resource value is missing. If impliedZeroForMissingResources is false (the default), the value of AVG (NUM_CPU) will be value 2 / 1 record = 2. If impliedZeroForMissingResources is true, the value of AVG (NUM_CPU) will be value 2 / 2 records = 1.

In effect, impliedZeroForMissingResources = "true" has the same effect as if the records were given as:

```
Record #1: User1, NUM_CPU=2,  MEM_USED=4
Record #2: User1, NUM_CPU=0, MEM_USED=4
```

CreateAccountRelationship:

The CreateAccountRelationship stage adds a client that represents an account in SmartCloud Cost Management. The stage also associates the newly created client with a rate table and a user group in SmartCloud Cost Management, creating the user group if it does not already exist. The stage updates existing client accounts, so they can be associated with other user groups and it can also modify the rate table that is used.

Settings

The CreateAccountRelationship stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateAccountRelationship> element:

- active = "true | false": Setting this to true activates this stage within a job file. The default setting is true.
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- stopOnStageFailure = "true | false": Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Parameters

The following attributes can be set in the <Parameter> element:

- createGroup="true | false": Setting this to true means that the user group will be created if it does not exist. Setting this to false means that the user group will not be created if it does not exist. The default setting is true.

Mandatory Identifiers

The following attributes are expected in the CSR input data:

- CLIENTNAME: The client name that represents the account code. The client name is formed from the account code. If the client name already exists, then it can be updated for description, user group and rate table.
- ACCOUNTCODE: The account code that defines the structure, which reflects the chargeback hierarchy for the client.

Optional Identifiers

The following attributes are optional in the CSR input data:

- USERGROUPID: The id of the user group that the client is associated to. If the client account exists and it is not already associated, then the user groups associated with the client account are updated to this new group. A client account must be associated to at least one user group so that the client account is viable in SmartCloud Cost Management.
- USERGROUPDESC: Description of the user group that the client account is associated to. If USERGROUPID is specified and USERGROUPDESC is not, then the USERGROUPID value is used by default.
- RATETABLE: The rate table that the client account uses. If it is not specified, then the default STANDARD rate table is used. If the client account already exists, then the existing client rate table that is used is updated to this new table.
- DEFAULTACS: The default account code structure that the user group is associated with. If it is not specified, then the default Standard account code structure is used.

Example 1: New client account, user group does not exist

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,21,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_AT,2014-03-07 11:11:08,VM_LAUNCHED_AT,20
ACCOUNTCODE,Admins ApplicationA              ,
CLIENTNAME,ApplicationA,USERGROUPID,ApplicationA Admins,USERGROUPDESC,Admin group for ApplicationA,5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DISK,0
```

If the CreateAccountRelationship stage appears as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
```

```
      <Parameter exceptionProcess="true"/>
      <Parameter createGroup="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- The client account "ApplicationA" is created.
- User group "ApplicationA Admins" is created and client account "ApplicationA" is associated with it.
- Client account uses "STANDARD" rate table.

Example 2: Existing client account, user group exists and rate table specified

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,22,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_AT,2014-03-07 11:11:08,VM_LAUNCHED_A
ACCOUNTCODE,Admins ApplicationA                    ,
CLIENTNAME,ApplicationA,USERGROUPID,ApplicationA User,USERGROUPDESC,User group for ApplicationA,RATETABLE,CUSTOM, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,V
```

If the CreateAccountRelationship stage appears as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
      <Parameter createGroup="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- Client account "ApplicationA" association is updated with the user group ApplicationA User.
- Client account "ApplicationA" is updated to use the CUSTOM rate table.

Considerations when using the CreateAccountRelationship stage

Consider the following when using the CreateAccountRelationship stage:

- The exception file can contain records for input data, where the user group does not exist and createGroup is set to false.
- The exception file can contain records for input data, where the STANDARD rate table does not exist in SmartCloud Cost Management when creating a user group with the default rate table.
- The exception file can contain records for input data, where the STANDARD account code structure does not exist in SmartCloud Cost Management when creating a user group with the default account code structure.
- The exceptionProcess parameter must be turned on for records that are written to the exception file. For more information, see the related topic in the Configuration guide.
- If CLIENTNAME is not specified in the CSR record, then the CSR record is added to the exception file.
- If ACCOUNTCODE is not specified in the CSR record, then the CSR record is added to the exception file.

CreateIdentifierFromIdentifiers:

The `CreateIdentifierFromIdentifiers` stage creates a new identifier by compounding the strings that comprise others. This can be done by taking whole identifier strings or by taking substrings.

Settings

The CreateIdentifierFromIdentifiers stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateIdentifierFromIdentifiers>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the new composite identifier created from existing identifiers.

FromIdentifiers:

The following attributes can be set in the `<FromIdentifier>` element:

- `name = "identifier_name"`: The name of an identifier that will be used to form part of the new identifier. The order of the `FromIdentifier` elements defines the order of concatenated values that appear in the new identifier value.
- `offset="numeric_value"`: This value, in conjunction with `length` allows you to set how much of the original identifier is used in creating the new identifier. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- `length="numeric_value"`: This value, in conjunction with `offset` allows you to set how much of the original identifier is used in creating the new identifier. If you want to use five characters of an identifer, the length is set to 5. This is set to 50 by default.
- `delimiter="delimiter_value"`: If set this value will be concatenated to the end of the FromIdentifier. This is null by default.

The following is an exmaple of a `FromIdentifier` setting:

```
<FromIdentifier name="User" offset="1" length="5" delimiter="a"/>
```

Parameters

The following attributes can be set in the `<Parameter>` element:

- keepLength="true | false" : The parameter keepLength specifies whether the entire length should be included. If the length specified is longer than the identifier value, the value is padded with spaces to meet the maximum length. The default for this setting is "false".
- modifyIfExists=="true | false": If this parameter is set to true, and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

A CreateIdentifierFromIdentifiers stage can be created as follows:

```
<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>
```

Running the preceding CreateIdentifierFromIdentifiers stage creates the following output CSR file.

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User, /
"joe",Account_Code,"joeaSrvr1b",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User, /
"mary",Account_Code,"maryaSrvr2b",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,/
"joan",Account_Code,"joanaSrvr3b",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,/
"joan",Account_Code,"joanaSrvr3b",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,/
"joe",Account_Code,"joeaSrvr1b",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.

CreateIdentifierFromRegEx:

The `CreateIdentifierFromRegEx` stage creates a new identifier the value of which is derived using a regular expression. The regular expression will derive the new value using an existing value.

Settings

The CreateIdentifierFromRegEx stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateIdentifierFromRegEx>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the new composite identifier created from existing identifiers.

FromIdentifier:

The following attributes can be set in the `<FromIdentifier>` element:

- `name ="identifier_name"`: The name of an identifier that will be used to form part of the new identifier.
- `regEx="regular_expression"`: The regular expression used to parse the identifier value.
- `value="regular_expression_group"`: The value of the segment that will be used to create the new identifier. Please see the example for more information.

Parameters

The following attributes can be set in the `<Parameter>` element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joe.allen@xyz.com",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"mary.kay@xyz.com",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joan.jet@xyz.com",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joan.jet@xyz.com",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joe.allen@xyz.com",2,EXEMRCV,1,EXBYRCV,2817
```

If the `CreateIdentifierFromRegEx` stage appears as follows:

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

The output CSR file appears as follows:

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joe.allen@xyz.com", \
FirstName,"joe",LastName,"allen",FullName,"allen, joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"mary.kay@xyz.com", \
FirstName,"mary",LastName,"kay",FullName,"kay, mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joan.jet@xyz.com", \
FirstName,"joan",LastName,"jet",FullName,"jet, joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joan.jet@xyz.com", \
FirstName,"joan",LastName,"jet",FullName,"jet, joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joe.allen@xyz.com", \
FirstName,"joe",LastName,"allen",FullName,"allen, joe",2,EXEMRCV,1,EXBYRCV,2817
```

The identifiers `FirstName`, `LastName`, and `FullName` were added.

CreateIdentifierFromTable:

The `CreateIdentifierFromTable` stage creates a new identifier from the values defined in the conversion table.

Settings

The `CreateIdentifierFromTable` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateIdentifierFromTable>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)

- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the new composite identifier created from another identifier's value as a lookup to a conversion table. Only one new identifier can be specified in the stage.

FromIdentifier:

The following attributes can be set in the `<FromIdentifier>` element:

- `name = "identifier_name"`: The name of an identifier the value of which will be sought in the conversion table. If a match is not found in the conversion process, the Identifier will be added with a value of spaces unless the `exceptionProcess` is turned on. In that case, the record will be written to the exception file.
- `offset="numeric_value"`: This value, in conjunction with `length` allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- `length="numeric_value"`: This value, in conjunction with `offset` allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifer, the length is set to 5. This is set to 50 by default.

Files:

The following attributes can be set in the `<File>` element:

- `name = "file_name"`: This is set to the name of the file that contains the conversion table. The number of definition entries that you can enter in the conversion table is limited only by the memory available to Integrator.
- `type="table | exception"`: There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- `encoding="system | encoding_scheme"`: This option is available only if the `type` is set to `table`. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, e.g., UTF-8.
- `format="CSROutput | CSRPlusOutput"`: This option is available only if the `type` is set to `exception`. The exception file can be in any output format that is supported by 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.

DBLookups:

The `<DBLookup>` element over rides the default functionality of loading the conversion table from file and loads the conversion mappings from the SmartCloud Cost Management database instead. The following attributes can be set in the `<DBLookup>` element:

- process = "process_name": The name of the Process Definition corresponding to the conversion mappings that you want to reference. If this is not set, the Process Id of the job file is used as the default.
- discrimator="first | last | largest": When the discrimator attribute is set to "first", it references the first conversion entry that matches the usage period. When set to "last", it references the last conversion entry that matches the usage period. When set to "largest", it references the conversion entry which matches the largest timeline for the usage period. If this is not set, the default is last.
- cacheSize="integrer value between 1 and 99": This configures the maximum number of periods that may be cached in memory for processing the CSR input. Each distinct usage period in the CSR input necessitates a lookup in the database. These periods are cached for subsequent records. Setting the cacheSize to a high value will improve performance of the stage but will use more memory. Set to 24 by default.

Parameters:

The following attributes can be set in the <Parameter> element:

- exceptionProcess="true | false" : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry.
- sort="true | false" : If the parameter sort is set to true (the default and recommended), an internal sort of the conversion table is performed.
- upperCase="true | false" : The conversion table is case-sensitive. For convenience, you can enter uppercase values in the conversion table for your identifiers and then set the parameter upperCase="true". This ensures that identifier values in your CSR input data that are lowercase or mixed case are processed. The default for upperCase is false.
- writeNoMatch="true | false" : If this 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 setting is false.
- modifyIfExists=="true | false": If this parameter is set to true and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example - File

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",
Account_Code,"ATM",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,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`.

Example - DBLookup

Example 1: discriminator = "first"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the `CreateIdentifierFromTable` stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="first"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 1:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"John",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

Example 2: discriminator = "last"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the `CreateIdentifierFromTable` stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="last"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 2:

```
VVMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Pat",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

Example 3: discriminator = "largest"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the CreateIdentifierFromTable stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="largest"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 3:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",Account_Code,
"Paul",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Note: Refer to the sample jobfile SampleAutomatedConversions.xml when using this stage.

CreateIdentifierFromValue:

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

Settings

The CreateIdentifierFromValue stage accepts the following input elements.

Attributes:

The following attributes can be set in the <CreateIdentifierFromValue> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": The name of the new identifier.
- value=" value": The value that is attributed to the new identifier.

Parameters :

The following attributes can be set in the <Parameter> element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the CreateIdentifierFromValue stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,2817
```

The identifier Break_Room was added with a value of North.

CreateResourceFromConversion:

The CreateResourceFromConversion stage creates a new resource, with the value derived using an arithmetic expression.

Settings

The CreateResourceFromConversion stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateResourceFromConversion> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)

- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not exist in the table.
- `symbol="a-z"`: This allows you to pick a variable which can be used to represent the value of the new resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

FromResource

The following attributes can be set in the `<FromResource>` element:

- `name = "resource_name"`: The name of an existing resource with the value used to derive a new resource value.
- `symbol="a-z"`: This allows you to pick a variable which will be given the value of the named resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

Parameters

The following attributes can be set in the `<Parameter>` element:

- `formula = "arithmetic_expression"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element. In addition minimum and maximum capability is provided. Rounding functions are also provided based on the Java Rounding Modes definition.
- `modifyIfExists=="true | false"`: If this parameter is set to true and the identifier exists, the existing identifier value is modified with the specified value. If this is set to false (the default), the existing identifier value is not changed.

Example 1

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Total_Resource">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   <FromResource name="EXBYRCV" symbol="b"/>
   </FromResources>
   </Resource>
   </Resources>
   <Parameters>
```

```
   <Parameter formula="(a+b)/60"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Total_Resource,65.7
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Total_Resource,64.4
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Total_Resource,45.81667
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Total_Resource,50.23333
Example,20070117,20070117,00:00:00,23:59:59,1,2,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.

Example 2

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,3092,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,3000,EXBYRCV,2817
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Max_Resource">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   <FromResource name="EXBYRCV" symbol="b"/>
   </FromResources>
   </Resource>
   </Resources>
   <Parameters>
   <Parameter formula="max(a,b)"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Max_Resource,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Max_Resource,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Max_Resource,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,3092,EXBYRCV,3013,Max_Resource, 3092
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,3000,EXBYRCV,2817,Max_Resource, 3000
```

The resource `Max_Resource` was added. The value for the new resource is built from the maximum of the existing resource values.

Example 3

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",1,EXEMRCV,1.6
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",1,EXEMRCV,1.5
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"john",1,EXEMRCV,1.2
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Half_Up_Res">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   </FromResources>
   </Resource>
   </Resources>
```

```
   <Parameters>
   <Parameter formula="HALF_UP(a)"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1.6,Half_Up_Res,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1.5,Half_Up_Res,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"john",2,EXEMRCV,1.2,Half_Up_Res,1
```

The resource `Half_Up_Res` was added. The value for the new resource is built from the `HALF_UP` of the existing resource values.

Considerations for Using CreateResourceFromConversion

Consider the following when using the `CreateResourceFromConversion` stage:

- Rounding functionality supported based on the Java Rounding Modes definition as follows:
  - **`CEILING:`** rounding mode to round towards positive infinity.
  - **`DOWN:`** rounding mode to round towards zero.
  - **`FLOOR:`** rounding mode to round towards negative infinity.
  - **`HALF_DOWN:`** rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.
  - **`HALF_EVEN:`** rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor.
  - **`HALF_UP:`** rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.
  - **`UNNECESSARY:`** rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary.
  - **`UP:`** rounding mode to round away from zero.
- Minimum and maximum capability is on two Resource elements only and does not include arithmetic expressions in it.
- Rounding capability is on one Resource element only and does not include arithmetic expressions in it.

CreateResourceFromDuration:

The `CreateResourceFromDuration` stage creates a new resource the value of which is set by calculating the difference between the start time and the end time in a CSR or CSR+ record. These times can be taken from the time fields in the record (start date, end date, start time, end time) or they can be taken from the identifier values in the record. Once the new duration resource has been created, it can be used in subsequent Integrator stages by using a mathematical formula to modify other resources.

Settings

The `CreateResourceFromDuration` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateResourceFromDuration>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of the new resource. If the Duration calculation results in a value of 0 for the units specified, then no Resource is created. You must add the duration resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.

FromDateTimes:

The following attributes can be set in the `<FromDateTime>` element:

- `name = "identifier_name"`: The name of the new resource.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `units = "milliseconds | seconds | minutes | hours | days"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element. Duration calculations use truncation, so partial units get dropped. For example, if the CSR record shows a duration of 2.5 hours, and the unit specified is hours, the Duration resource will be written as “2”.
- `dataFormat=java_date_format` : This paramter provides the format which will be used to interpret the timestamp values. The parameter dateFormat uses the conventions described by Java's SimpleDateFormat class. See its javadoc for examples of how to specify a date format.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example 1: Creating a resource using the time fields

In this example, assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20080117,20080117,08:00:00,08:00:45,,1,Feed,Srvr1,2,EXEMRCV,1,EXBYRCV,3941
Example,20080117,20080117,09:00:00,09:30:00,,1,Feed,Srvr2,2,EXEMRCV,1,EXBYRCV,3863
Example,20080117,20080117,10:00:00,14:30:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,2748
Example,20080117,20080217,08:00:00,06:00:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,3013
```

If the `CreateResourceFromDuration` stage appears as follows:

```
<Stage name="CreateResourceFromDuration" active="true">
       <Resources>
          <Resource name="Duration">
          </Resource>
       </Resources>
       <Parameters>
          <Parameter units="seconds"/>
          <Parameter modifyIfExists="true"/>
       </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20080117,20080117,08:00:00,08:00:45,,1,Feed,Srvr1,2,EXEMRCV,1,EXBYRCV,3941,Duration,45
Example,20080117,20080117,09:00:00,09:30:00,,1,Feed,Srvr2,2,EXEMRCV,1,EXBYRCV,3863,Duration,1800
Example,20080117,20080117,10:00:00,14:30:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,2748,Duration,9000
Example,20080117,20080217,08:00:00,06:00:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,3013,Duration,79200
```

The start time and end time are calculated using the time fields in the CSR or CSR+ record. For example, in the first record, the start and end duration is 45 seconds. As the `units` parameter value is `"seconds"`, a resource named `Duration` was created with a value of `45`. In the second record, the start and end duration is 30 minutes. Therefore, the `Duration` resource value is `1800` seconds.

Example 2: Creating a resource using identifier values

In this example, assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20080117,20080117,08:00:00,08:00:45,,3,Feed,Srvr1,StartDate,"2007/08/19  16:00:00", \
EndDate," 2007/08/19  16:30:15",01,LLY202,1846
Example,20080117,20080117,09:00:00,09:30:00,,3,Feed,Srvr2,StartDate,"2007/08/19  19:00:00", \
EndDate," 2007/08/19  19:00:15",01,LLY202,1846
```

If the `CreateResourceFromDuration` stage appears as follows:

```
<Stage name="CreateResourceFromDuration" active="true">
       <Resources>
          <Resource name="UserSpecifiedDuration">
             <FromDateTimes>
                <FromDateTime name="StartDate"/>
                <FromDateTime name="EndDate"/>
             </FromDateTimes>
          </Resource>
       </Resources>
       <Parameters>
          <Parameter dateFormat="yyyy/MM/dd HH:mm:ss"/>
          <Parameter units="minutes"/>
          <Parameter modifyIfExists="true"/>
       </Parameters>
    </Stage>
```

The output CSR file appears as follows:

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20080117,20080117,08:00:00,08:00:45,1,3,Feed,"Srvr1",StartDate,"2007/08/19 16:00:00",EndDate, \
" 2007/08/19 16:30:15", 2,LLY202,1846,UserSpecifiedDuration,30
Example,20080117,20080117,09:00:00,09:30:00,,3,Feed,Srvr2,StartDate,"2007/08/19  19:00:00",EndDate, \
" 2007/08/19  19:00:15",01,LLY202,1846
```

In this example, the `FromDateTime` elements specify the identifier names to use for calculating the duration (`StartDate` and `EndDate`). The `dateFormat` parameter provides the format that will be used to interpret the timestamp values. In the first record, the duration is 30 minutes and 15 seconds and the `units` parameter is `"minutes"`, so a resource named `UserSpecifiedDuration` was created with a value of `30`. Duration calculations use truncation, so the partial units of 15 seconds are dropped.

In the second record, the duration is 15 seconds, and the units are specified as minutes. Because the units value is in minutes and values are truncated, the calculation results in a value of 0 and a `UserSpecifiedDuration` resource was not created in this record.

CreateResourceFromValue:

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

Settings

The `CreateResourceFromValue` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateResourceFromValue>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "resource_name"`: The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.
- `value=" numeric_value"`: The new resource value.

Parameters :

The following attributes can be set in the `<Parameter>` element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
```

The resource Num_Recs was added with a value of 1.

CreateUserRelationship:

A user is an individual with access rights to SmartCloud Cost Management web-based applications. Each user can belong to one or more user groups. Users are granted the rights and privileges that are granted to the group. The CreateUserRelationship stage adds a user and associates that user to a user group in SmartCloud Cost Management. The stage can also update existing users to be associated to other user groups.

Settings

The CreateUserRelationship stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateUserRelationship> element:

- active = "true | false": Setting this attribute to true activates this stage within a job file. The default setting is true.
- trace = "true | false" : Setting this attribute to true enables the output of trace lines for this stage. The default setting is false.
- stopOnStageFailure = "true | false": Setting this attribute to true halts the execution of this stage if an error occurs. The default setting is true.

Files

The following attributes can be set in the <File> element:

- name = "file_name": This is set to the name of the exception file.
- type = "exception": The type and then the format or encoding must be specified.
- format = "CSROutput | CSRPlusOutput": This option is only available if the type is set to exception. The exception file can be in any output format that is supported by Integrator. The format is defined by the stage name of the output type. For example, if the stage CSROutput or CSRPlusOutput is active, the exception file is produced as CSR or CSR+ file.

Mandatory Identifiers

The following attributes are expected in the CSR input data:

- USERID: The identifier of the user.

Optional Identifiers

The following identifiers are optional in the CSR input data:

- USERDESC: Description of the user which is used as full name. If USERDESC is not specified, then the USERID value is used by default.
- USERGROUPID: The ID of the user group that the client is associated to. If the user exists and the user is not already associated, then the user groups associated with that user are updated to this new group. If the user already exists in the database, you may want to associate other groups to that user. If however, you

don't specify a group and the user is already in the database, then it will have a group already associated to it. In this scenario, the record can be ignored.

•

Parameters

- `defaultUserGroup="<userGroupId>"`: The Id of the user group which is used as a default in some scenarios when group is not defined in CSR file.

Example 1: New user, user group exists

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,19,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATI
USERID,john.smith, USERGROUPID,ApplicationA Admins, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DISK,0
```

If the `CreateUserRelationship` stage is displayed as follows:

```
<Stage name="CreateUserRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- The user `john.smith` is created.
- User is now associated to the `ApplicationA Admins` user group.

Example 2: Existing user, user group exists

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,19,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATI
USERID,john.smith, USERGROUPID,ApplicationB Admins, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DISK,0
```

If the `CreateUserRelationship` stage is displayed as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- User `john.smith` is now associated to the `ApplicationB Admins` user group.

Considerations when using the `CreateUserRelationship` stage

Consider the following when using the `CreateUserRelationship` stage:

- If you are creating clients and user groups with users from this `CreateUserRelationship`, the stage must be called after the `CreateAccountRelationship` stage.
- If the user is associated to an `admin` group in the stage, it is disassociated from all previous groups as a consequence.
- The `exceptionProcess` parameter must be turned on for records that are written to the exception file. For more information, see the *Enabling exception processing* topic in the Configuration guide.

- If no user group is specified and the user does not exist, then add a CSR record to the exception file.
- If no user group is specified and the user exists, then ignore the CSR record.
- If no user group is specified, the default user group is configured, and the user exists, then ignore the CSR record.
- If `USERID` is not specified in the CSR record, then the CSR record is added to the exception file.

CSROutput:

The `CSROutput` stage produces a CSR file.

Parameters:

The following attributes can be set in the `<parameter>` element:

- `<Parameter keepZeroValueResources="true | false"/>`: This parameter when set to true, enables resources with zero values to be written to CSR files and billing output or read from CSR files and billing output. Resources with zero values are normally discarded.

Example

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

In this example, the CSR `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.

Parameters:

The following attributes can be set in the `<parameter>` element:

- `<Parameter keepZeroValueResources="true | false"/>`: This parameter when set to true, enables resources with zero values to be written to or read from CSR files and in billing output. Resources with zero values are normally discarded.

Example

```
<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.

DropFields:

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

Settings

The DropFields stage accepts the following input elements.

Attributes:

The following attributes can be set in the <DropFields> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Fields:

The following attributes can be set in the <Field> element:

- name = "resource_name | identifier_name": This sets the name of the resource or identifier to drop.

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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the DropFields stage appears as follows:

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

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"mary",1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",1,EXBYRCV,2817
```

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

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.

Settings

The DropIdentifiers stage accepts the following input elements.

Attributes:

The following attributes can be set in the <DropIdentifiers> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": This sets the name of the identifier to drop.

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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817
```

If the DropIdentifiers stage appears as follows:

```
<Stage name="DropIdentifiers" active="true">
   <Identifiers>
   <Identifier name="Feed">
   </Identifiers>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,2817
```

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

DropResources:

The `DropResources` stage drops a specified resource from a 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.

Settings

The `DropResources` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<DropResources>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: This sets the name of the resource to drop.

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.

Note: 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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817
```

If the `DropResources` stage appears as follows:

```
<Stage name="DropResources" active="true">
   <Resources>
   <Resource name="Feed"/>
   </Resources>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,2817
```

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

ExcludeRecsByDate:

The `ExcludeRecsByDate` stage excludes 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.

Settings

The `ExcludeRecsByDate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<ExcludeRecsByDate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the `<Parameter>` element:

- `fromDate = "from_date"`: This parameter allows you to specify the exclusion start date. The date is entered in the format YYYYMMDD.
- `toDate= "to_date"`: This parameter allows you to specify the exclusion end date. The date is entered in the format YYYYMMDD.
- `keyWord= "**PREDAY | **CURDAY | **RNDATE | **PREMON | **CURMON | **PREWEK | **CURWEK"`: This parameter allows you to use a date keyword to specify the exclusion date range.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `ExcludeRecsByDate` stage appears as follows:

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

Or

```
<Stage name="ExcludeRecsByDate" active="true">
   <Parameters>
   <Parameter fromDate="20070101"/>
   <Parameter toDate="20070131"/>
   </Parameters>
</Stage>
```

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

```
Example,20070217,20070217,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

Only those records with end dates in February are included.

ExcludeRecsByPresence:

The `ExcludeRecsByPresence` stage drops records based on the existence or non-existence of identifiers, resources, or both.

Settings

The `ExcludeRecsByPresence` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<ExcludeRecsByPresence>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"` : This parameter allows you to specify the resource that will be tested for existence and excluded based on the following condition.
- `exists="true | false"`: Setting this parameter to true will cause a record to be dropped if it contains the named resource. Setting it to false will cause a record to be dropped if it does not contain the resource.

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name" : This parameter allows you to specify the identifier that will be tested for existence and excluded based on the following condition.
- exists="true | false": Setting this parameter to true will cause a record to be dropped if it contains the named identifier. Setting it to false will cause a record to be dropped if it does not contain the identifier.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple entries are treated as OR conditions. If any one of the conditions is met, the record is dropped.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",2,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,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,1,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.

ExcludeRecsByValue:

The ExcludeRecsByValue stage drops records based on an identifier values, resource values, or both.

Settings

The ExcludeRecsByValue stage accepts the following input elements.

Attributes:

The following attributes can be set in the <ExcludeRecsByValue> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)

- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"` : This setting allows you to enter the name of a resource for comparison. If the following comparison conditions are met, the record containing this identifier will not be included in the output.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. 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)
  - LIKE (starts with, ends with, and contains a string. Starts with the value format = `"string%"`, ends with the value format = `"%string"`, and contains the value format = `"%string%"`)
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is excluded from the output file.

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"` : This setting allows you to enter the name of an identifier for comparison. If the following comparison conditions are met, the record containing this identifier will not be included in the output.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. The comparison conditions have been stated previously.
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is excluded from the output file.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. 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 dropped.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `ExcludeRecsByValue` stage appears as follows:

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

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,
EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,
EXBYRCV,3863
```

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

FormatDateIdentifier:

The `FormatDateIdentifier` stage allows you to reformat a date type identifier.

Settings

The `FormatDateIdentifier` stage accepts the following input elements:

Attributes:

The following attributes can be set in the `<FormatDateIdentifier>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Identifiers:

The following attribute can be set in the `<Identifier>` element:

- name = `"identifier_name"`: The name of the date identifier to be reformatted.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `inputFormat = "java date_time pattern"`: This parameter specifies the format of the input date identifier.
- `outputFormat = "java date_time pattern"`: This parameter specifies the format of the output date identifier.

Example

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

```
Example,20070602,,10:00:00,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI00001,Jobname,ALIREC6,
Start_date,20070602,Shift,1,20,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,1.31,Z005,1708,Z006,1367,
Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,1333,Z012,10270,Z013,25987,
Num_Rcds,5
```

```
Example,20070602,,11:47:56,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,BLI00002,Jobname,ABCDLYBK,
Start_date,20070602,Shift,1,19,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,Z005,93160,Z006,
4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,Z011,258557,Z012,494924,
Z013,7950768,Num_Rcds,1

Example,20070602,,11:40:25,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,CLI00003,Jobname,ABCOPER,
Start_date,20070602,Shift,1,17,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,13,Z006,13,
Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284,Num_Rcds,4
```

If the `FormatDateIdentifier` stage appears as follows:

```
<Stage name="FormatDateIdentifier" active="true" trace="false" >
 <Identifiers>
 <Identifier name="Start_date"/>
 </Identifiers>
 <Parameters>
 <Parameter inputFormat="yyyyMMdd"/>
 <Parameter outputFormat="dd.MM.yyyy"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070602,20070602,10:00:00,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"ALI00001",
Jobname,"ALIREC6",Start_date,"02.06.2007",Shift,"1",19,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,
1.31,Z005,1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,
1333,Z012,10270,Z013,25987

Example,20070602,20070602,11:47:56,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"BLI00002",Jobname,
"ABCDLYBK",Start_date,"02.06.2007",Shift,"1",18,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,
Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,Z011,258557,
Z012,494924,Z013,7950768

Example,20070602,20070602,11:40:25,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"CLI00003",Jobname,
"ABCOPER",Start_date,"02.06.2007",Shift,"1",16,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,
13,Z006,13,Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284
```

The format of the identifier `Start_date` was changed to `dd.MM.yyyy` in the output CSR file.

IdentifierConversionFromTable:

The `IdentifierConversionFromTable` stage converts an identifiers value using the identifiers own value or another identifiers value as a lookup to a conversion table.

Settings

The `IdentifierConversionFromTable` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<IdentifierConversionFromTable>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the identifier created by converting the value of the identifier using the identifiers own value or another identifiers value as a lookup to a conversion table. If the identifier defined for conversion is not found in the input record, the record is treated as an exception record. Only one new identifier can be specified per stage.

FromIdentifier

The following attributes can be set in the <FromIdentifier> element:

- name = "identifier_name": The name of an identifier the value of which will be sought in the conversion table. If a match is not found in the conversion process, the Identifier will be added with a value of spaces unless the exceptionProcess is turned on. In that case, the record will be written to the exception file.
- offset="numeric_value": This value, in conjunction with length allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- length="numeric_value": This value, in conjunction with offset allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifier, the length is set to 5. This is set to 50 by default.

Files

The following attributes can be set in the <File> element:

- name = "file_name": This is set to the name of the file that contains the conversion table. The number of definition entries that you can enter in the conversion table is limited only by the memory available to Integrator.
- type="table | exception": There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- encoding="system | encoding_scheme": This option is available only if the type is set to table. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, for example, UTF-8.
- format="CSROutput | CSRPlusOutput": This option is available only if the type is set to exception. The exception file can be in any output format that is supported by 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.

DBLookups

The <DBLookup> element over rides the default functionality of loading the conversion table from file and loads the conversion mappings from the SmartCloud Cost Management database instead. The following attributes can be set in the <DBLookup> element:

- process = "process_name": The name of the Process Definition corresponding to the conversion mappings that you want to reference. If this is not set, the Process Id of the job file is used as the default.
- discrimator="first | last | largest": When the discrimator attribute is set to "first", it references the first conversion entry that matches the usage period. When set to "last", it references the last conversion entry that matches the usage period. When set to "largest", it references the conversion entry which matches the largest timeline for the usage period. If this is not set, the default is last.
- cacheSize="integrer value between 1 and 99": This configures the maximum number of periods that may be cached in memory for processing the CSR input.

Each distinct usage period in the CSR input necessitates a lookup in the database. These periods are cached for subsequent records. Setting the `cacheSize` to a high value will improve performance of the stage but will use more memory. Set to `24` by default.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `exceptionProcess="true | false"` : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry. Records will not be written to the exception file.
- `sort="true | false"` : If the parameter sort is set to "true" (the default and recommended), an internal sort of the conversion table is performed.
- `upperCase="true | false"` : 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".
- `writeNoMatch="true | false"` : If this 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 setting is false.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example - File

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoe",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerMary",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,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`.

Example - DBLookup

Example 1: discriminator = "first"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the `IdentifierConversionFromTable` stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="first"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 1:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"John",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

Example 2: discriminator = "last"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,
"VM1",Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the IdentifierConversionFromTable stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="last"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 2:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Pat",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Example 3: discriminator = "largest"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,
"VM1",Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the IdentifierConversionFromTable stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="largest"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
```

```
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 3:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Paul",2, VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

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.
- 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 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.
- 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"`.
- 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 exists, the existing identifier value is modified with the specified value. If `modifyIfExists="false"` (the default) existing identifier value is not changed.
- If the parameter `discrimator` is set to..
  - `last`(the default), it references the last conversion entry that matches the usage period.
  - `first`, it references the first conversion entry that matches the usage period.
  - `largest`, it references the conversion entry which matches the largest timeline for the usage period.

- 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 (fore example, "`,,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 Integrator.

Note: Refer to the sample jobfile `SampleAutomatedConversions.xml` when using this stage.

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.

Settings

The `IncludeRecsByDate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByDate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the `<Parameter>` element:

- `fromDate = "from_date"`: This parameter allows you to specify the inclusion start date. The date is entered in the format YYYYMMDD.
- `toDate= "to_date"`: This parameter allows you to specify the inclusion end date. The date is entered in the format YYYYMMDD.
- `keyWord= "**PREDAY | **CURDAY | **RNDATE | **PREMON | **CURMON | **PREWEK | **CURWEK"`: This parameter allows you to use a date keyword to specify the inclusion date range.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `IncludeRecsByDate` stage appears as follows:

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

Or

```
<Stage name="IncludeRecsByDate" active="true">
   <Parameters>
   <Parameter fromDate="20070101"/>
   <Parameter toDate="20070131"/>
   </Parameters>
</Stage>
```

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

```
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
```

Only those records with end dates in January are included.

IncludeRecsByPresence:

The `IncludeRecsByPresence` stage includes records based on the existence or non-existence of identifiers, resources, or both. The resource or identifier name can also be defined using regular expressions, so you can use wildcards and not need an exact match for the identifier or resource name.

Settings

The `IncludeRecsByPresence` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByPresence>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name_regular_expression"` : This parameter allows you to specify either the exact resource name or a regular expression for the resource that will be tested for existence and included based on the following condition.
- `exists = "true | false"`: Setting this parameter to true will cause a record to be included if it contains the named resource. Setting it to false will cause a record to be dropped if it contains the named resource.

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name_regular_expression" : This parameter allows you to specify either the exact identifier name or a regular expression for the identifier that will be tested for existence and included based on the following condition.
- exists = "true | false": Setting this parameter to true will cause a record to be included if it contains the named identifier. Setting it to false will cause a record to be dropped if it contains the named identifier.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple entries are treated as OR conditions. If any one of the conditions is met, the record is included.

Example 1

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,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.

Example 2

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

```
Example,20070602,,10:00:00,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI00001,Jobname,
ALIREC6,Start_date,20070602,Shift,1,20,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,1.31,Z005,
1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,1333,
Z012,10270,Z013,25987,Num_Rcds,5

Example,20070602,,11:47:56,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,BLI00002,Jobname,
```

```
ABCDLYBK,Start_date,20070602,Shift,1,19,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,
Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,
Z011,258557,Z012,494924,Z013,7950768,Num_Rcds,1

Example,20070602,,11:40:25,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,CLI00003,Jobname,ABCOPER,
Start_date,20070602,Shift,1,17,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,13,Z006,13,
Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284,Num_Rcds,4
```

If the `IncludeRecsByPresence` stage appears as follows:

```
<Stage name="IncludeRecsByPresence" active="true" trace="true" >
   <Resources>
   <Resource name="[ZZ][0-9][0-9].*" exists="true">
   </Resources>
 </Stage>
```

The output CSR file appears as follows:

```
The output CSR file appears as follows:
Example,20070602,20070602,10:00:00,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"ALI00001",
Jobname,"ALIREC6",Start_date,"20070602",Shift,"1",19,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,
Z033,1.31,Z005,1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,
Z010,14876,Z011,1333,Z012,10270,Z013,25987

Example,20070602,20070602,11:47:56,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"BLI00002",
Jobname,"ABCDLYBK",Start_date,"20070602",Shift,"1",18,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,
62.3,Z033,66.6,Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,
Z010,753696,Z011,258557,Z012,494924,Z013,7950768

Example,20070602,20070602,11:40:25,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"CLI00003",
Jobname,"ABCOPER",Start_date,"20070602",Shift,"1",16,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,
Z033,0.3,Z005,13,Z006,13,Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284
```

The three records in the input file were included because they contain a resource which matches the regular expression i.e. ZZ followed by 2 or more digits.

IncludeRecsByValue:

The `IncludeRecsByValue` stage includes records based on identifier values, resource values, or both. If the comparison is true, the record is included.

Settings

The `IncludeRecsByValue` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByValue>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"` : This setting allows you to enter the name of a resource for comparison. If the following comparison conditions are met, the record will be included in the output. If a field specified for inclusion contains a blank value, the record is included.

- `cond="GT | GE | EQ |LT | LE | LIKE"` : This setting allows you to enter the comparison condition. 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)
  - LIKE (starts with, ends with, and contains a string. Starts with the value format = `"string%"`, ends with the value format = `"%string"`, and contains the value format = `"%string%"`)
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is included in the output file.

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"` : This setting allows you to enter the name of an identifier for comparison. If the following comparison conditions are met, the record will be included in the output. If a field specified for inclusion contains a blank value, the record is included.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. The comparison conditions have been stated previously.
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is included in the output file.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is included. If a field specified for inclusion contains a blank value, the record is included.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `IncludeRecsByValue` stage appears as follows:

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

The output CSR file appears as follows:

```
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan", 2,EXEMRCV,1,EXBYRCV,2748
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan", 2,EXEMRCV,1,EXBYRCV,3013
>Example,20070117,20070117,00:00:00,23:59:59,1,2,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 included.

MaxRecords:

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

Settings

The MaxRecords stage accepts the following input elements.

Attributes:

The following attributes can be set in the <MaxRecords> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the <Parameter> element:

- number = "numeric_value": This parameter allows you to specify the maximum number of records that can be processed.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
```

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

PadIdentifier:

The PadIdentifier stage allows you to pad an identifier with a specified character, either to the left or the right of the identifier.

Settings

The PadIdentifier stage accepts the following input elements:

Attributes:

The following attributes can be set in the PadIdentifier element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Identifiers:

The following attribute can be set in the <Identifier> element:

- name = "identifier_name": This parameter specifies the identifier to be padded.

Parameters:

The following attributes can be set in the <Parameter> element:

- length = "numeric_value" : This parameter specifies the length that the identifier should be.
- padChar = "any_char" : This parameter specifies the pad character to use. The default is 0.
- justify = "left | right" : This parameter specifies left or right justification for the identifier, prior to padding.

Note: Justification specifies how to justify the identifier before executing the padding, therefore justifying the identifier to the right, means the padding will take place to the left of the identifier value.

Example

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

```
WinDisk,20100603,20100603,00:00:00,23:59:59,,3,Feed,Server1-C,Path,C:,Folder,C:,2,DISKFILE,9,
DISKSIZE,3.290056

WinDisk,20100603,20100603,00:00:00,23:59:59,,3,Feed,Server1-C,Path,"C:\Program Files",Folder,
"Program Files",2,DISKFILE,25335,DISKSIZE,4.145787
```

If the PadIdentifier stage appears as follows:

```
<Stage name="PadIdentifier" active="true">
 <Identifiers>
 <Identifier name="Folder"/>
 </Identifiers>
 <Parameters>
 <Parameter length="20"/>
```

```
 <Parameter padChar="?"/>
 <Parameter justify="right"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
WinDisk,20100603,20100603,00:00:00,23:59:59,1,4,Feed,"Server1-C",Path,"C:",Folder,
"????????????????????C:",Account_Code,"????????????????????C:",2,DISKFILE,9,DISKSIZE,
3.290056

WinDisk,20100603,20100603,00:00:00,23:59:59,1,4,Feed,"Server1-C",Path,"C:\Program Files",
Folder,"???????Program Files",Account_Code,"???????Program Files",2,DISKFILE,25335,
DISKSIZE,4.145787
```

The identifier `Folder` has a length of 20 and is padded with the ? character. The justification is set to right so the identifier is right justified and the padding is performed to the left of the identifier.

Prorate:

This process prorates incoming data based on a proration table and a set of proration parameters.

Settings

The `Prorate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<Prorate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Files:

The following attributes can be set in the `<File>` element:

- `name = "file_name"`: This is set to the name of the file that contains the proration table.
- `type="prorationtable | exception"`: There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- `encoding="system | encoding_scheme"`: This option is available only if the `type` is set to `table`. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, e.g., UTF-8.
- `format="CSROutput | CSRPlusOutput"`: This option is available only if the `type` is set to `exception`. The exception file can be in any output format that is supported by 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.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `IdentifierName`: Name of identifier field to search.
- `IdentifierStart = "numeric_value"`: First position in field to check. Default is 1.
- `IdentifierLength = "numeric_value"`: Number of characters to compare. Default is the entire field.
- `Audit = "true | false"`: Indicates whether or not to write original fields as audit trail. Default is true.
- `AllowNon100Totals = "true | false"`: - Indicates whether or not total proration percentages must equal 100 percent. The default is true.
- `exceptionProcess="true | false"` : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry. Records will not be written to the exception file.
- `NewIdentifier = "identifier_name"`: New identifier field name to assign to the updated field. If not specified, the original name will be used.
- `CatchallIdentifier="identifier_name"` : Identifier to be used if there is no match for the identifier field in the proration table. If no value is entered, the default is Catchall. You may specify more than one catchall parameter. If no catchall parameters are specified, catchall processing will not be used.
  - `CatchallPercent="numeric_value"`: Percentage to be used. Default is 100.
  - `CatchallRate="resource_name"`: Rate code to be prorated. Default is all rate codes.

For a detailed example of the Prorate stage, see the *Prorating resources* section.

RenameFields:

The `RenameFields` stage renames specified identifiers and resources.

Settings

The `RenameFields` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<RenameFields>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Fields:

The following attributes can be set in the `<Field>` element:

- `name = "resource_name | identifier_name"`: Set this to the name of the existing resource or identifier that you would like to rename.

- newName="string_value": Set this to the new resource or identifier name.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the RenameFields stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",UserName,"joe",2,Emails,1,Bytes,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",UserName,"mary",2,Emails,1,Bytes,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",UserName,"joan",2,Emails,1,Bytes,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",UserName,"joan",2,Emails,1,Bytes,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",UserName,"joe",2,Emails,1,Bytes,2817
```

- The field User was renamed UserName.
- The field EXEMRCV was renamed Emails.
- The field EXBYRCV was renamed Bytes.

RenameResourceFromIdentifier:

The RenameResourceFromIdentifier stage allows you to use an identifier value as the value of a resource (rate code).

Settings

The RenameResourceFromIdentifier stage accepts the following input elements:

Attributes:

The following attributes can be set in the <RenameResourceFromIdentifier> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Resources:

The following attribute can be set in the <Resource> element:

- name = "resource_name" : This parameter allows you to specify the resource that will be renamed.

Identifiers:

The following attribute can be set in the <Identifier> element:

- name = "identifier_name" : This parameter specifies the identifier to be used for the renaming.

Parameters :

The following attribute can be set in the <Parameter> element:

- dropIdentifier = "true | false" : The parameter dropIdentifier specifies whether the identifier should be included in the output CSR file or not. If the parameter is set to true, the identifier will be dropped and not included in the output CSR file. If the parameter is set to false, the identifier will be included in the output CSR file.
- renameType = "prefix | suffix | overwrite": The parameter renameType specifies how the resource is renamed. If the parameter is set to prefix, the name of the resource is started with the identifier value. If the parameter is set to suffix, the name of the resource is ended with the identifier value. If the parameter is set to overwrite (default setting), the name of the resource is replaced with the identifier value.

Example

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

```
WinDisk,20100602,20100602,00:00:00,23:59:59,,3,Feed,Server1-C,Path,C:,Folder,C:,2,DISKFILE,
11,DISKSIZE,1.998663

WinDisk,20100602,20100602,00:00:00,23:59:59,,3,Feed,Server1-C,Path,"C:\Program Files",Folder,
"Program Files",2,DISKFILE,16379,DISKSIZE,3.404005
```

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources>
 <Parameters>
 <Parameter dropIdentifier="false"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+2010060220100602024C:
",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,Feed,"Server1-
C",Path,"C:",Folder,"C:",Account_Code,"C:
",2,C:,11,DISKSIZE,1.998663

"CSR+2010060220100602024Program Files
",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,Feed,"Server1-
C",Path,"C:\Program Files",Folder,"Program
Files",Account_Code,"Program Files ",2,C:\Program
Files,16379,DISKSIZE,3.404005
```

The resource DISKFILE has been renamed using the identifier Path.

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources>
```

```
 <Parameters>
 <Parameter dropIdentifier="false"/>
 <Parameter renameType="prefix"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+2010060220100602024C: ",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,
Feed,"Server1- C",Path,"C:",Folder,"C:",Account_Code,"C: ",2,C:DISKFILE,11,
DISKSIZE,1.998663
"CSR+20100602201006020240Program Files ",WinDisk,20100602,20100602,00:00:00,
23:59:59,1,4,Feed,"Server1- C",Path,"C:\Program Files",Folder,"Program Files",
Account_Code,"Program Files ",2,C:\Program FilesDISKFILE,16379,DISKSIZE,3.404005
```

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources> <Parameters>
 <Parameter dropIdentifier="false"/>
 <Parameter renameType="suffix"/>
</Parameters> </Stage>
```

The output CSR file appears as follows:

```
"CSR+2010060220100602024C: ",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,
Feed,"Server1- C",Path,"C:",Folder,"C:",Account_Code,"C: ",2,DISKFILEC:,
11,DISKSIZE,1.998663
"CSR+20100602201006020240Program Files ",WinDisk,20100602,20100602,00:00:00,
23:59:59,1,4,Feed,"Server1- C",Path,"C:\Program Files",Folder,"Program Files",
Account_Code,"Program Files ",2,DISKFILEC:\Program Files,16379,DISKSIZE,3.404005
```

ResourceConversion:

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

Settings

The ResourceConversion stage accepts the following input elements.

Attributes:

The following attributes can be set in the <ResourceConversion> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the <Resource> element:

- name = "resource_name": The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.

- `symbol="a-z"`: This allows you to pick a variable which can be used to represent the value of the new resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

FromResource:

The following attributes can be set in the `<FromResource>` element:

- `name = "resource_name"`: The name of an existing resource the value of which will be used to derive a new resource value.
- `symbol="a-z"`: This allows you to pick a variable that will be given the value of the named resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `formula = "arithmetic_expression"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `ResourceConversion` stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,60,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,60,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,60,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,60,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,60,EXBYRCV,2817
```

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

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.

Settings

The Sort stage accepts the following input elements.

Attributes:

The following attributes can be set in the <Sort> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": This allows you to set the identifier on which the sort is based..
- length="numeric value": 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 a length is specified and the length of the field is less than the specified length, blanks will be used to pad out the length.

Parameters:

The following attributes can be set in the <Parameter> element:

- Order="Ascending | Descending": This allows you to set the sort type. The default order is ascending.

Note: This process is memory dependent. If there is not enough memory to do the sort, the process will take a long time to complete.

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,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.

TierResources:

The `TierResources` stage allows tiering of a list of rate codes or all applicable rate codes. A list of resources (rate codes) can be explicitly specified in the XML file. Alternatively, the list of resources (rate codes) can remain empty and the stage determines what resources (rate codes) should be tiered and tiers them.

Overview

Tiered pricing allows you to charge different rates for resource usage based on predefined thresholds. The daily processing of feeds does not change. You continue to process all of your daily feeds as before, with the only change being that the rate values of the proposed tier rates are set to 0. At the end of the accounting period, most commonly a month, a job file can be run to perform the tier pricing for each rate. The basic flow is as follows:

- Extract the summary records for the period using the Universal Collector called DATABASE. The identifiers are `Account_Code` and `RateCode`. The resource must be defined with the name `Units`.
- Run an `Integrator Aggregation` stage to aggregate the `Account_Code`, `RateCode`, and `Units` for each summary record.

  Note: The summary record contains 1 resource only.
  After this stage, a CSR file is created with just `Account_Code`, `RateCode`, and `Units`.
- Run an Integrator `IncludeRecsByValue` stage to drop all of the rates that are not tier rates. You could move this stage before the Aggregation stage if required. After this stage, the CSR file contains just the rate codes that are tier rates.
- Run the Integrator `TierResources` stage. This stage tiers one or more rates at a time.
- Run the standard Bill step to cost the data.
- Run the standard `DBLoad` step to load the costed summary and detail data.

If you decide to change a tier rate, delete the loads for this job and rerun. Set up each tier rate code as described in the requirements section below. It is not important to determine the actual rate value that you want to change, since the reprocessing of this data can be completed by deleting the load with the incorrect data and rerunning.

Requirements

- For resources (rate codes) that you want to tier, you must set the rate values for the parent rates to 0. You do not want to cost the rate during daily processing. The summary records are written to the DB with usage information, but with no cost information.
- You must define the child tier rates with the rate values that you want to charge for in the tier. For example:
  - Set `Z001` parent rate value to 0.

  - Add a child tier rate `Z001_1` with a rate value of `2.00`.
  - Add a child tier rate `Z001_2` with a rate value of 1.8 (10% discount for volume).
  - Add another child tier rate `Z001_3` with a rate value of 1.6.
- The rate pattern used must also be set on the parent tier rate.
- Threshold and Percent values must also be defined on the child tier rates. Percent values are only applicable to rates with the rate pattern "Tier Monetary Individual (%)" and "Tier Monetary Highest (%)".
- A tier rate for the highest threshold must be defined with an empty threshold value, this rate will be used for values above the highest threshold.

Settings

The `TierResources` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<TierResources>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. The default setting is true.
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- `stopOnStageFailure = "true | false"`: Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Resources

The following attributes can be set in the `<Resources>` element:

- `name = "resource_name"`: The name of the resource or rate code used for tiering. Only resources specified under the `<Resources>` element are tiered by this stage. If no resources are specified under the `<Resources>` element, all applicable tiered rates defined in the Rate Tables panel are used.

Example 1: Rate Pattern ="Tier Individual" or "Tier Monetary Individual (%)"

Assume that the following data exists for the rate codes you want to tier.

```
Z001 = 158
Z001 = 300
Z001 = 66
```

And the following thresholds and percentages are defined on the Tiered Rates of Z001:

Table 49. Defined thresholds

RateCode	Threshold	Percentages
Z001_1	100	10
Z001_2	200	15
Z001_3	300	20
Z001_4		25

If the `TierResources` stage appears as follows:

```
<Stage name="TierResources" active="true">
 <Resources>
 <Resource name="Z001"/>
 </Resources>
</Stage>
```

The resource units are tiered as follows, where `Z001_orig` has the original values:

```
Z001_orig,158,Z001_1,100,Z001_2,58
Z001_orig,300,Z001_1,100,Z001_2,100,Z001_3,100
Z001_orig,66,Z001_1,66
```

When using the "Tier Monetary Individual (%)", rate pattern, the Bill step applies the percentage to the resource units at each tier.

Example 2: Rate Pattern ="Tier Highest" or "Tier Monetary Highest (%)"

Assume that the following data exists for the rate codes you want to tier.

```
Z001 = 158
Z001 = 300
Z001 = 66
```

And the following thresholds and percentages are defined on the Tiered Rates of Z001:

Table 50. Defined thresholds

RateCode	Threshold	Percentages
Z001_1	100	10
Z001_2	200	15
Z001_3	300	20
Z001_4		25

If the `TierResources` stage appears as follows:

```
<Stage name="TierResources" active="true">
 <Resources>
 <Resource name="Z001"/>
 </Resources>
</Stage>
```

The resource units are tiered as follows, where `Z001_orig` has the original values:

```
Z001_orig,158,Z001_2,158
Z001_orig,300,Z001_3,300
Z001_orig,66,Z001_1,66
```

When using the "Tier Monetary Highest (%)" rate pattern, the Bill step applies the percentage to the resource units at the corresponding tier.

Additional example

The `<TUAM_home_dir>/samples/jobfiles/SampleTierResources.xml` job file contains an example that shows the process described in this topic.

Related concepts:

Universal Collector overview
The Universal Collector within Integrator was designed to extend the input stage of the integrator and simplify the creation of a new collector.

TierSingleResource:

The `TierSingleResource` stage allows tiering of a single rate code.

Note: This stage has been deprecated and replaced with the new `TierResources` stage. See the related concept topic for more information about the `TierResources` stage. When migrating from this stage to the `TierResources` stage, tiered rates must be redefined manually using the Rate Tables panel. For more information about defining rate tables, see the related *Administering the system guide*.

Overview

Tiered pricing allows you to charge different rates for resource usage based on predefined thresholds. The daily processing of feeds does not change. You continue to process all of your daily feeds as before, with the only change being setting the rate values of the proposed tier rates to 0.

At the end of the accounting period (most commonly a month), a jobfile can be run to perform the tier pricing for each rate. The basic flow is as follow:

- Extract the summary records for the period – using the Integrator Generic Collector – DB. The identifiers will be Account Code and Rate Code. The resource will have the name Units.
- Run an Integrator Aggregation step to aggregate the account code, rate code and units for each summary record. Remember, the summary record only contains 1 resource – so after this step, you will have a CSR file with just account codes, rate codes and units.
- Run an Integrator `IncludeRecsByValue` step to drop all of the rates that are not tier rates. You could move this step before the Aggregation step if you want. After this step, you will have a CSR file with just the rate codes that are tier rates.
- Run one or more Integrator `TierSingleResource` steps. This step will tier one rate at a time. So if you have multiple rates that you want to tier, you must run this step for each rate.
- Run the standard Bill step to cost the data.
- Run the standard DBLoad step to load the costed summary and detail data.

If you decide to change a tier rate, delete the loads for this job and rerun.

Set up each tier rate code as described in the following section. It is not important to determine the actual rate value that you want to change, since the re-processing of this data is easy to do.

Requirements

- Resources (rate codes) that you want to Tier – you must set the rate values for those rates to 0. You do not want to cost the rate during your daily processing. The summary records will be written to the DB with usage information, but with no cost information.
- Resources (rate codes) that you want to Tier, can only be 6 bytes long. If you want to tier a default rate that is longer than 6 bytes, you must rename that resource in an Integrator step. This is because, the process will dynamically add –n to the rate code for each tier. For example, if the tier rate code is Z001, then the tier 1 rate will be Z001-1 and the tier 2 rate will be Z001-2, and so on.
- You must define the tier rates with the rate values that you want to charge for the tier. So again, in the previous example, set Z001 rate value to 0, add a rate

Z001-1 with a rate value of say 2.00 and add rate Z001-2 with a rate value of say 1.8 (10% discount for volume) and add rate Z001-3 with a rate value of 1.6.

Settings

The `TierSingleResource` stage accepts the following input elements:

Attributes:

The following attributes can be set in the `<TierSingleResource>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `thresholds = "n {,n}":` This parameter allows you to specify the threshold values to be used for tiering.
- `ratecode = "rate_code":` This parameter allows you to specify the rate code to be used for tiering.
- `method = "HighestTier | IndividualTier":` This parameter allows you to specify the tiering option to be used. With HighestTier, where ever the resource units fall, all of the units will be charged at that rate. With IndividualTier (the default), the units will be charged in their respective tier. For example, if the thresholds are 10,20,30 and the units are 25, then 10 will be charged at tier 1, 10 at tier 2 and 5 at tier 3.

Note: There is always one more tier than number of tiers specified on the thresholds. For example, if you set thresholds 10, 20, 30 – then there are 4 tiers: tier 1 is 0-10, tier 2 is > 10-20, tier 3 is > 20-30 and tier 4 is everything over 30.

Example

Assume that the following data exists for the rate code that you want to Tier:

```
ResourceUnits = 158
ResourceUnits = 300
ResourceUnits = 66
```

If the TierSingleResource stage appears as follows:

```
<Stage name="TierSingleResource" active="true">
 <Parameters>
 <Parameter thresholds="100,200,300"/>
 <Parameter ratecode="ABC101"/>
 <Parameter method="IndividualTier"/>
 </Parameters>
</Stage>
```

The resource units will be tiered as follows:

```
ResourceUnits,158, ABC101-1,100, ABC101-2,58
ResourceUnits,300, ABC101-1,100, ABC101-2,100, ABC101-3,100
ResourceUnits,66, ABC101-1,66
```

If the TierSingleResource stage appears as follows:

```
<Stage name="TierSingleResource" active="true">
 <Parameters>
 <Parameter thresholds="100,200,300"/>
 <Parameter ratecode="ABC101"/>
 <Parameter method="HighestTier"/>
 </Parameters>
</Stage>
```

The resource units will be tiered as follows:

```
ResourceUnits,158, ABC101-2,158
ResourceUnits,300, ABC101-3,300
ResourceUnits,66, ABC101-1,66
```

UpdateConversionFromRecord:

The `UpdateConversionFromRecord` stage inserts and/or updates lookup records in the SmartCloud Cost Management database conversion table based on the process definition for a usage period. The `UpdateConversionFromRecord` stage uses an identifier for the source of the lookup and an identifier for the target of the lookup. Modifications to the conversion records for a process definition only occurs after the `lockdown` date of the process definition.

Settings

The `UpdateConversionFromRecord` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<UpdateConversionFromRecord>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. The default setting is true.
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- `stopOnStageFailure = "true | false"`: Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The identifier name used as the target for a conversion record in the conversion table. Only one target identifier can be specified for each stage.

FromIdentifier

The following attributes can be set in the `<FromIdentifier>` element:

- `name = "identifier_name"`: The name of the first identifier used as the lookup or low identifier to a conversion record in the conversion table. If a match is found for this identifier in the conversion table for the usage period, then the expiry date of the existing record is set to the period immediately before the usage period start date. A new record is then added to the conversion table where the effective date is set to the usage period start date and the expiry date is set to the default value of `Dec 31, 2199`. If no match is found for this identifier in the conversion table for the usage period, a new record is added to the conversion table where the effective date is set to the usage period start date and the expiry date is set to the default value of `Dec 31, 2199`.

- `name = "identifier_name"`: The name of the second identifier used as the lookup or high identifier to the conversion record in the conversion table. Only used for the addition of new conversion records, for example, `allowNewConversionEntries=true`. If used when conversion updates are allowed, for example, `allowConversionEntryUpdate=forwardOnly`, then the conversion record is written to an exception file.
- `offset="numeric_value"`: This value, in conjunction with length allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you use the whole identifier string, your offset is set to 1. However, if you are selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- `length="numeric_value"`: This value, in conjunction with offset allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifier, the length is set to 5. This is set to 50 by default.

Files

The following attributes can be set in the `<File>` element:

- `name = "file_name"`: This is set to the name of the exception file.
- `type="exception"`: To allow records to be written to an exception file.
- `format="CSROutput | CSRPlusOutput"`: This option is only available if the `type` is set to `exception`. The exception file can be in any output format that is supported by Integrator. The format is defined by the stage name of the output type. For example, if the stage `CSROutput` or `CSRPlusOutput` is active, the exception file is produced as CSR or CSR+ file.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `process="<process_name>"`: The name of the process definition corresponding to the conversion mappings that you want to insert or update. If this is not set, the `Process Id` of the job file is used as the default.
- `useUsageEndDate="true | false"`: Setting this to true means that the usage period end date is used for lookup of the conversion records whose expiry date is after the usage period start date. Setting this to false means that usage period start date is used for lookup of the conversion records whose expiry date is after the usage period start date. The default setting is false.
- `allowNewConversionEntries="true | false"`: Setting this to true means that if there are no conversion records to update, it inserts a new conversion record into the conversion table. Setting this to false means that it writes the conversion record to an exception file. The default setting is true.
- `allowConversionEntryUpdate= "forwardOnly | none"`: Determines how conversion records are updated. Updates are not allowed if a high identifier is specified. The configuration is as follows:
  - `forwardOnly (default):` Updates can only occur in time order whereby the expiry date of the last conversion record is updated and a new conversion record is added.
  - `none`: Does not allow updates and write the conversion records to an exception file. Not affected if high identifier is specified.

- `strictLockdown="true"`: Setting this to true means that if you try to update or insert a conversion record before the Process Definition lock down date, then the conversion record is written to an exception file. The default setting is true.
- `dayBoundary="true"`: Setting this to true means that the conversion records added to the conversion table will have a start date with a time which is the beginning of that day, i.e. midnight. Setting this to false means that the conversion records added to the conversion table will have a start date with a time which is the start time of the CSR record. The default setting is false.

Example 1: allowNewConversionEntries="true"

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

```
VMWARE,20120314,20120314,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM3",User,"Sean",2,VMCPU,4,VMMEM,2048
```

If the `UpdateConversionFromRecord` stage appears as follows:

```
<Stage name="UpdateConversionFromRecord" active="true">
   <Identifiers>
   <Identifier name="User">
   <FromIdentifiers>
   <FromIdentifier name="VMName" offset="1" length="4"/>
   </FromIdentifiers>
   </Identifier>
   </Identifiers>
   <Parameters>
   <Parameter process="VMWARE"/>
   <Parameter useUsageEndDate="false"/>
   <Parameter allowNewConversionEntries="true"/>
   </Parameters>
</Stage>
```

And the conversion table appears as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
```

The updated conversion table appears as follows:

```
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
'VMWARE','VM3',,'2012-03-14 00:00:00','2199-12-31 23:59:59','Sean'
```

The value for the `VMName` identifier and the start date of the usage period was used to determine the new record with new target mappings in the conversion table for a process definition.

Example 2: allowConversionEntryUpdate="forwardOnly"

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

```
VMWARE,20120111,20120111,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",User,"Paul",2,VMCPU,2,VMMEM,2048
VMWARE,20120116,20120116,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",User,"Pat",2,VMCPU,2,VMMEM,2048
```

If the `UpdateConversionFromRecord` stage appears as follows:

```
<Stage name="UpdateConversionFromRecord" active="true">
   <Identifiers>
   <Identifier name="User">
   <FromIdentifiers>
   <FromIdentifier name="VMName" offset="1" length="4"/>
   </FromIdentifiers>
   </Identifier>
   </Identifiers>
   <Parameters>
   <Parameter process="VMWARE"/>
```

```
   <Parameter useUsageEndDate="false"/>
   <Parameter allowConversionEntryUpdate="forwardOnly"/>
   </Parameters>
</Stage>
```

And the conversion table appears as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
```

The updated conversion table appears as follows:

```
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-15 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-16 00:00:00','2199-12-31 23:59:59','Pat'
```

The value for the `VMName` identifier and the start date of the usage period was used to determine the new expiry date for existing records in the conversion table, as well as new records with new target mappings in the conversion table for a process definition.

Considerations for using UpdateConversionFromRecord

Consider the following when using the `UpdateConversionFromRecord` stage:

- Only one new identifier can be specified in the stage.
- The number of definition entries that you can enter in the conversion table is limited by the memory available to the Integrator.
- The exception file can contain records for input data that was before the `lockdown` date of the process definition or anything else that violates the rules for adding new conversion records.
- The first `<FromIdentifier>` is low identifier and the second is high identifier.
- High identifier cannot be specified for updates to conversion records for the existing identifier in the conversion table.
- Process definition is automatically added to the database if it does not exist.
- The `exceptionProcess` parameter must be turned on for records to be written to the exception file. For more information, see the *Enabling exception processing* topic in the Configuration guide.
- If using multiple `UpdateConversionFromRecord` stages within a job step, then each stage must reference a distinct process definition.

Note: Refer to sample jobfile `SampleAutomatedConversions.xml` when using this stage.

Running job files

You can run job files in batch mode.

You can run job files in either of the following ways:

- **In batch (recommended).** Using Job Runner, job files can be run in batch mode on a defined schedule (daily, monthly, and so on) to convert usage metering data created by your system into CSR or CSR+ files. Once the CSR or CSR+ files are created, the data in the files is processed and the output is loaded into the database using the SmartCloud Cost Management Processing Engine.

Running job files in batch mode

Running job files in batch mode enables you to run the files on a defined schedule (daily, monthly, and so on). To run job files in batch mode, use the `<SCCM_install_dir>/bin/startJobRunner.sh` file if you are running the files on a Linux platform .

Scheduling the job files

To run a job file in batch mode, you can use any batch scheduling tool. Specify the file that you want to run in the scheduling tool and provide the job file name followed by any optional parameters:

- **Linux :** `startJobRunner.sh <job file name>`

Optional Parameters

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

`-jf <job filter> -pf <process filter> -sf <step filter> -date`

Where:

`-jf` = 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.

`-pf` = 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.

`-sf` = 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` = a date keyword or start date and stop date. This parameter specifies the date for the data that you want to collect. If you do not provide a date, the default date is the previous day. This is the equivalent of using the date keyword PREDAY. If you provide a start date, but no stop date, the default stop date is CURDAY.

Examples

Job Runner Command	Description
`startJobRunner.sh Nightly.xml`	Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml.
`startJobRunner.sh Nightly.xm -jf Nightly`	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.
`startJobRunner.sh Nightly.xml -jf Nightly -pf All -sf DatabaseLoad`	Job Runner runs only the active DatabaseLoad steps in all processes in the Nightly job. No other steps in the job file are run.

Job Runner Command	Description
`startJobRunner.sh Nightly.xml -jf All -pf All -sf All -date 20080604`	Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the LogDate parameter 20080604.
`startJobRunner.sh Nightly.xml -jf All -pf All -sf All -date RNDATE`	Job Runner runs all active jobs, processes, and steps in the job file Nightly.xml using the date parameter RNDATE.

Passing the Date Parameter from the Command Line

If you need to use a date parameter other than PREDAY, for example you want to process and backload old log files, include the date parameter at the command line when you schedule Job Runner. When you enter a date parameter that includes a date range, such as CURMON, Job Runner runs the data collection process for each day in the range. If log file generation and email messaging is enabled in the job file, a separate log file and email message is generated for each day.

Passing the Date Parameter from the Job File

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

- **You are running a snapshot collector (DBSpace, Windows Disk, or Exchange Server Mailbox).** Snapshot data collectors collect data that is current as of the date and time that the collectors are run. 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 date parameter value. For example, if you use the date 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, use the keyword RNDATE as the date parameter. When RNDATE is specified in the job file, you must ensure that the command line does not include a date parameter or that RNDATE is provided at the command line. Date values provided in the command line will override values in the job file.
- **You are running the Transactions collector.** The Transactions collector uses the date parameters CURMON, PREMON, or the date/period in yyyypp format only. The yyyypp format is specific to the Transactions collector and cannot be passed from the command line. In addition, CURMON and PREMON cannot be passed from the command line for the Transactions collector.

Viewing job file logs

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. You can view log files by date in the Administration Console. The default directory for job log files is `<SCCM_install_dir>/logs/jobrunner`. If you do not want to use the default directory, you must define the alternative directory in the configuration properties file.

About this task

To view job file logs, complete the following steps:

Procedure

1. In Administration Console, click **Tasks** > **Log Files**.

2. On the Log Files page, select the ID of the job for the log files that you want to view in the drop-down list. All log files for the job ID that you select are displayed hierarchically by date.
3. Click the **Expand** icon to the left of the dates for the logs that you want to view. You can expand or collapse the nodes in the log file tree using the **Expand** and **Collapse** icon. Icons help you to quickly determine whether a job, process, or step within the file completed successfully (green), completed with warnings (yellow), or failed (red).

Viewing sample job files

SmartCloud Cost Management includes sample job files that you can modify for your organization. The default directory for sample job log files is `<SCCM_install_dir>/samples/jobfiles`. SmartCloud Cost Management includes sample job files that you can modify for your organization.

Setting accounting dates

SmartCloud Cost Management input records contain a *usage* start date and end date that specify the date range in which the resources in the record were consumed. The Bill program uses the usage end date to calculate *accounting* start and end dates, which are used for billing and reporting. The accounting start and end dates may be the same as or different than the usage end date.

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

- **CIMSBill Detail file.** The accounting dates are in the `ACCOUNTING-START-DATE` and `ACCOUNTING-END-DATE` fields.
- **Summary file.** The accounting dates are in the `AccountingFromDate` and `AccountingToDate` fields (starting positions 167 and 175, respectively). The `Period` and `Year` fields also reflect the accounting end dates. (These fields are at starting positions 282 and 284, respectively.) For example, if the end date is `20080916` and the date falls within the 9th period in the CIMSCalendar table, the `Period` field would contain `09` and the `Year` field `2008`.

Account codes and account code conversion

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

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

Both programs, Integrator and Acct, can perform account code conversion. The preferred program is Integrator, using the process stage `IdentifierConversionFromTable`. Integrator is far more powerful and can provide not only account code conversion, but can also perform conversion on any identifier value in the CSR file or CSR+ file.

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

Setting up account codes and performing account code conversion

This section describes the account codes structure and how to define and convert the account code using the parameters in the job file.

Defining the account code structure

One of the first steps in implementing SmartCloud Cost Management is deciding on a final account code structure. The structure is the chargeback hierarchy for the organization. The levels of this hierarchy are arranged from highest to lowest level within the account code.

The account code can consist of up to 128 characters. For example, your final account code structure could consist of:

DDDCCCCUUUUUUUU

D = Division (three characters) (highest level)

C = Cost Center (four characters)

U = User (eight characters) (lowest level)

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

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

Note: For detailed information on how to set the account code structure , see the section *Setting the account code structure*.

Defining the account code

The account code is derived from an identifier or identifiers in the input file.

You must specify the identifier(s) that you want to use to define the account code as follows:

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

– The input file contains an Account_Code identifier and other identifiers that you want to use. In this case, you must define the Account_Code identifier in addition to the other identifiers.

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

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

Defining the account code input field using the Integrator program:

To define the account code input field in the job file, use the FromIdentifiers section in the process stage IdentifierConversionFromTable.

The following example illustrates how to define the identifiers that you want:

```
<FromIdentifiers>
   <FromIdentifier name=" UserName " offset="1" length="10"/>
   <FromIdentifier name=" Division " offset="1" length="2"/>
</FromIdentifiers>
```

In this example, the identifiers that will be used are UserName and Division. The identifiers offset and length to use are specified. If you want to use the full value for an identifier, use an offset of 1 and the maximum length of the identifier value as defined in the input file.

For example, assume that the maximum value for UserName identifier in the input file is 10 characters and that the input file contains three UserName values, KSMITH, GBONIFANT, and CGREENHALL. If you set the offset to 1 and the length to 10 and the entire identifier values would be used. If you want to use a portion of the identifier value, use an offset position at which you want to start the value and the corresponding length. Using the UserName example, if you set the offset at 2 and length at 8, the identifier values used would be SMITH, BONIFANT, and GREENHALL. You can define as many from identifiers as you want. The maximum number of characters for an input field is 127.

Related concepts:

"Integrator job file structure" on page 97
This section provides the complete set of integrator account code conversion options.

Defining the account code input field using the Acct program:

To define the account code input field in the job file, use the ACCOUNT FIELD control statement parameter to define the identifiers that you want to use.

Begin with Field0 and continue sequentially as needed as shown in the following example:

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

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

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

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

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

You can define up to 10 account fields (Field0-Field9). The maximum number of characters for an account field is 127.

Note: Although the maximum number of characters for each account field is 127, the overall length of all account fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values cannot exceed 127 bytes.

Related reference:

"Parameter element" on page 79
This topic provides a complete listing of the parameter options available for the Acct program.

Converting account code input field values to uppercase

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

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

Use the one of the following control statement parameters to convert lowercase account code input field values to uppercase values in the resulting account code:

- UPPERCASE ACCOUNT FIELDS (Acct)
- upperCase="true" (Integrator)

For example, the value ddic would be converted to DDIC. By using this option Acct or Integrator account code processing becomes case-insensitive and makes defining account conversion tables much easier.

Setting the account code conversion options (Acct program)

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

Enabling account code conversion:

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

Defining the identifier values for account code conversion:

Use the `DEFINE FIELD` control statement parameter in the job file to specify the offset and length of the account code input field values that you want to use for conversion.

Begin at `Field0` and continue sequentially as needed. Note that if you created an account code input field, the define fields correlate to the account field values, not the original identifiers in the input file.

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

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

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

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

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

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

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

You can define up to 10 define fields (`Field0-Field9`). The maximum number of characters for an account field is 127.

Note: Although maximum number of characters for each define field is 127, the overall length of all define fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

Defining optional move fields:

Use the optional `DEFINE MOVEFIELD` control statement parameter in the job file to include either the define field values or a literal in the output account code.

Begin at `Field0` and continue sequentially as needed. You can define up to 10 move fields (`Field0-Field9`). Each move field can be a maximum of 127 characters.

Note: Although the maximum number of characters for each move field is 127, the overall length of all move fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

To define a define field as move field:

If you specify a define field as the move field, the define field value will be included at the beginning, end, or within the account code as defined in the account code conversion table. For example assume that you have defined define fields as follows:

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

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

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

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

To define a literal as move field:

If you specify a literal as the move field, the literal value will be included at the beginning, end, or within the account code as defined in the account code conversion table. For example, if you want to define the literal `abc` as a move field, the control statement parameter would be as follows:

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

Defining the account code conversion table:

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

For example:

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

Or

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

Related concepts:

"Creating the account code conversion table (Acct program)" on page 170
This topic describes how to create the account code conversion table.

Enabling exception processing:

Exception processing instructs the system to write all records that do not match an entry in the account code conversion table to an exception file.

To enable exception file processing, include the control statement parameter `EXCEPTION FILE PROCESSING ON` in the job file as follows:

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

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

Considerations for exception processing

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

Creating the account code conversion table (Acct program):

The account code conversion table is an ASCII text file that contains the definitions required to convert the identifier values defined by the account code input field to user defined output account codes.

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

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

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

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

Considerations when creating an account code conversion table

Consider the following when creating the account code conversion table:

- The account code conversion table is case-sensitive. For convenience, you can enter uppercase values in the table and use the `UPPERCASE ACCOUNT FIELDS` control statement parameter. This is especially helpful it you are using one

account code conversion table for multiple process definitions. This ensures that account code input field values that are lowercase or mixed case are processed.

- Each record in the table can be up to 385 characters (129 for the low identifier, 129 for the high identifier, and 127 for the new account code).
- The identifier fields follow these rules:
  - Each low and high identifier field is compared to the corresponding account code input field values. If the compares are true, the account code is assigned.
  - The low identifier fields are padded with x'00' and the high value fields are padded with x'FF'.
  - The high identifier field is set equal to the low field plus the high padding when the high identifier value is null.
  - You can enter up to 10 levels of 127 characters each for the low and high identifier fields. Each level must be delimited by a colon (:).

    For example, if you type AA as the low identifier and CC as the high identifier, all account codes that begin with characters greater than or equal to AA or less than or equal to CC are converted to the specified account code. If you type AAAAAAAA:BB as the low identifier and CCCCCCCC:DD as the high identifier, the Account Code lookup for the first level will match characters greater than or equal to AAAAAAAA or less than or equal to CCCCCCCC for the first identifier and the Account Code lookup for the second level will match characters greater than or equal to BB or less than or equal to DD for the second identifier. This process is repeated for each level/identifier specified.

    The asterisk (*) and the question mark (?) characters can be used as wildcard characters for both the low and high identifiers. Using the wildcard characters can degrade performance; use them only when needed.

    Note: Although the maximum number of characters for each field level is 127, the overall length of all levels added together cannot exceed 127 characters.
- You can include a default account code as the last entry in the account code conversion table by leaving the low and high identifier fields empty (for example, ",,DEFAULTCODE"). In this case, all records that contain identifier values that do not match an entry in the account code conversion table will be matched to the default code.
- If you do not include a default account code, input file records that contain identifier values that do not match an entry in the account code conversion table are written to the Acct output CSR+ file with the unconverted account code input field value as the account code.

  If you want unmatched records to go to an exception file for reprocessing at a later time, you need to use the `EXCEPTION FILE PROCESSING ON` control statement parameter. If you enable exception file processing, make sure that you have not included a default account code entry in the account code conversion table.
- If the `PRINT ACCOUNT NO-MATCH` statement is included in the control file, a message is written to the trace file showing the identifier values from the input file that were not matched during account code conversion. If this statement is not present, only the total number of unmatched records is provided. The system prints up to 1000 messages.
- The number of definition entries that you can enter in the table is limited only by the memory available to Acct.

Account code conversion example (Acct program):

This topic provides and example of an account code conversion.

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

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

DDDCCCCUUUUUUUU

D = Division (three characters) (highest level)

C = Cost Center (four characters)

U = User (eight characters) (lowest level)

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

Table 51. User identifier values

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

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

Define the account code and conversion parameters in the job file

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

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

Because you also want to include the define field value at the end of the account code, you need to define a move field for the value, also with an offset of 1 and a length of 7. Note that the `ACCOUNT CODE CONVERSION SORT` control statement parameter is required to enable account code conversion. The `EXCEPTION FILE PROCESSING ON` control statement parameter instructs the Acct program to produce an exception file.

Create the account code conversion table

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

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

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

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

Advanced account code conversion example (Acct program):

This topic provides an example of account code conversion.

Initial setup

The CurrentCSR.txt input file for "Acct" contains two key/value pairs that are used the following examples:

```
PROJID,"sasquatch"
UserName,"laura"
```

Example 1: No conversion

This example shows how to use the Acct program to create an account code without the use of a conversion table.

1. Account code conversion is commented out in the jobfile:
   ```
   <!--Parameter ControlCard="Account Code Conversion"/-->
   ```
2. Two Account Fields are defined:
   ```
   <Parameter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
   <Parameter ControlCard="ACCOUNT FIELD1,PROJID,1,8"/>
   ```
3. The resulting AcctCSR.txt output file contains the ACCOUNT_CODE key/value pair that looks like this:
   ```
   ACCOUNT_CODE,"laura     sasquatc"
   ```

Example 2: Conversion enabled, move is not enabled:

This example shows how to use the Acct program to create an account code using a conversion table.

1. The CurrentCSR.txt input file is the same as the one used in the initial setup.
2. The Acct code conversion table file (Accttabl.txt) contains these lines:

```
aura,aurab,aura2aurab_@1_@0
aurac,auraz,aurac2auraz_@0
b,,bb_@0_@1
d,f,ee_@0_@1_ee
l,p,llll_@0
m,,mmmmmmmm
o,,oo_@2_@0_oo
r,,rr_@2
s,,sss
t,,@0_@2_@1_ttt
u,,uuuuuuu_@0_@2
v,,vvvv
A,C,@1_AAA_@0
B,,@0_@1_BBBB
C,,CCCCCC
D,F,EE_@1_@0_EE
L,,@0_LLLL
O,,OO
R,,@2_RR
S,,SSSSSSSS
T,,TTT_@0_@2_@1
U,,@0_@2_UUUUUUU
V,,V
W,,WW
X,,XXX
Y,,YYYY
Z,,ZZZZZ
,,NOMATCH
```

3. Account code conversion is turned on in the jobfile (no sorting).

   ```
   <Parameter ControlCard="Account Code Conversion"/>
   ```

4. Account and Define Fields are set as follows:

   ```
   <Parameter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
   <Parameter ControlCard="ACCOUNT FIELD1,PROJID,1,8"/>
   <Parameter ControlCard="DEFINE FIELD0,2,5"/>
   <Parameter ControlCard="DEFINE FIELD1,11,3"/>
   ```

5. "Move" is NOT turned on:

   ```
   <!--Parameter ControlCard="DEFINE MOVEFLD0,1,10"/-->
   ```

6. The AcctCSR.txt (output file) ACCOUNT_CODE key/value pair looks like this:

   ```
   ACCOUNT_CODE,"aura2aurab__"
   ```

This is due to:

1. ACCOUNT FIELDs define initial string: `"laura sasquatc"`
2. DEFINE FIELDs set the string that is used to search the conversion table:
   a. DEFINE FIELD0,2,5: Start at 2nd character of string ("laura sasquatc") and use 5 characters ("aura ").
   b. DEFINE FIELD1,11,3: Start at 11th character of string ("laura sasquatc") and use 3 characters ("asq").
   c. The resulting string is used to search the conversion table: "aura asq" In this case, the account code conversion table entry is:

      ```
      aura,aurab,aura2aurab_@1_@0
      ```

   d. Move is not enabled, so Acct converts the initial "aura asq" account code to an ACCOUNT_CODE value of:

      ```
      aura2aurab__
      ```

      This is the account code conversion table's account code minus any "@0", "@1", etc characters.

Example 3: Conversion and move enabled

This example shows how to use the Acct program to create an account code using a conversion table and move fields.

1. The CurrentCSR.txt input file is the same as the one used in the initial setup.
2. The Acct code conversion table file (Accttabl.txt) is the same as the one used in example 2.
3. Conversion is turned on (no sorting) in the jobfile:

   ```
   <Parameter ControlCard="Account Code Conversion"/>
   ```

4. Account and Define Fields are defined as follows:

   ```
   <Parameter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
   <Parameter ControlCard="ACCOUNT FIELD1,PROJID,1,8"/>
   <Parameter ControlCard="DEFINE FIELD0,2,5"/>
   <Parameter ControlCard="DEFINE FIELD1,11,3"/>
   ```

5. Move processing is turned on in the jobfile, using the following control card values:

   ```
   <Parameter ControlCard="DEFINE MOVEFLD0,1,10"/>
   <Parameter ControlCard="DEFINE MOVEFLD1,,,abc"/>
   <Parameter ControlCard="DEFINE MOVEFLD2,11,8"/>
   ```

6. The AcctCSR.txt (output file) ACCOUNT_CODE key/value pair looks like this:

   ```
   ACCOUNT_CODE,"aura2aurab_abc_laura"
   ```

   This is due to:

   a. ACCOUNT FIELDs define initial string: "laura sasquatc"
   b. DEFINE FIELDs set the string that is used to search the conversion table:
      1) DEFINE FIELD0,2,5: Start at 2nd character of string ("laura sasquatc") and use 5 characters ("aura ").
      2) DEFINE FIELD1,11,3: Start at 11th character of string ("laura sasquatc") and use 3 characters ("asq").
      3) Resulting string: "aura asq"
      4) Acct uses this string ("aura asq") to determine the account conversion table entry. In this case, the account code conversion table entry is:

         ```
         aura,aurab,aura2aurab_@1_@0
         ```

      5) Move is enabled, so final ACCOUNT_CODE value for this line is the account code with the values for the "@0" ("laura") and "@1" ("abc") substituted into the string:

         ```
         aura2aurab_abc_laura
         ```

Related reference:

"Parameter element" on page 79
All Acct settings used in the above examples are documented under the Parameter element.

Setting the account code conversion options (Integrator program)

To perform account conversion, regardless of whether you are using the `Account_Code` identifier or another identifier, you need to set the conversion options described in the Integrator process stage `IdentifierConversionFromTable` in the job file and create an account code conversion table. Account codes are assigned by matching identifier values in the input file records to the values in the account code conversion table.

Enabling account code conversion:

To enable account code conversion, use the Integrator process stage `IdentifierConversionFromTable`.

Defining the identifier values for account code conversion:

The offset and length of the account code input field are implied when using the Integrator program. The values are set in the Integrator element `FromIdentifiers`.

For a comparison on the difference of how to specify these values in Acct and Integrator consider the following examples.

Example

Acct:

```
<Parameter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
<Parameter ControlCard="ACCOUNT FIELD1,Division,1,2"/>
<Parameter ControlCard="DEFINE FIELD0,1,10"/>
<Parameter ControlCard="DEFINE FIELD1,11,2"/>
```

Integrator:

```
Integrator:
<FromIdentifiers>
   <FromIdentifier name=" UserName " offset="1" length="10"/>
   <FromIdentifier name=" Division " offset="1" length="2"/>
</FromIdentifiers>
```

Example

Acct:

```
<Paramter ControlCard="ACCOUNT FIELD0,UserName,1,10"/>
<Paramter ControlCard="ACCOUNT FIELD1,Division,1,2"/>
<Paramter ControlCard="DEFINE FIELD0,1,8"/>
<Paramter ControlCard="DEFINE FIELD1,12,1"/>
```

Integrator:

```
<FromIdentifiers>
   <FromIdentifier name=" UserName " offset="1" length="8"/>
   <FromIdentifier name=" Division " offset="2" length="1"/>
</FromIdentifiers>
```

Defining optional move fields:

Use the optional MOVEFIELD definition in the job file to include either the move field values or a literal in the output account code.

Begin at Move0 and continue sequentially as needed. You can define up to 10 move fields (Move0-Move9). Each move field can be a maximum of 127 characters.

Note: Although the maximum number of characters for each move field is 127, the overall length of all move fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

To define MoveField:

If you specify a field as the move field, the move field value will be included at the beginning, end, or within the account code as defined in the account code conversion table. For example:

```
<MoveField name="MOVE0" offset="4" length="5" />
<MoveField name="MOVE1" offset="1" length="3" />
```

Note: The name given to the MoveField parameter must contain, at minimum, the value 0 and proceed sequentially up to a max value of 9. Each MoveField defined may be referenced in the Account Code conversion table with the syntax @x, for example @0 for MOVE0, @1 for MOVE1, and so on. More than one MoveField can be referenced in each entry and literals can be interspersed, for example @0JOE@2.

Example:

Assume that the Account Code conversion table is defined as follows:

```
romy,,John@0
u797,,Joe@1
```

Assume that the following CSR file is the input:

```
EXCHANGE,20070601,20070601,00:00:04,00:00:04,,2,"Feed","Server1","User","romydario@yahoo.es"
,4,EXEMRCV,0,EXBYRCV,0,EXEMSNT,1,EXBYSNT,366438

EXCHANGE,20070601,20070601,00:00:04,14:03:05,,2,"Feed","Server1","User","u797991@ntmcon02.emn.com"
,4,EXEMRCV,15,EXBYRCV,610456,EXEMSNT,0,EXBYSNT,0
```

If the MoveField definition appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true" trace="false" stopOnStageFailure="true" >
 <Identifiers>
  <Identifier name="Account_Code">
   <FromIdentifiers>
    <FromIdentifier name="User" offset="1" length="12"/>
   </FromIdentifiers>
   <MoveFields>
    <MoveField name="MOVE0" offset="4" length="5" />
    <MoveField name="MOVE1" offset="1" length="3" />
   </MoveFields>
  </Identifier>
 </Identifiers>
 <Files>
  <File name="AcctConv.txt" type="table" />
  <File name="MyException.txt" type="exception" format="CSROutput" />
 </Files>
 <Parameters>
  <Parameter exceptionProcess="true" />
  <Parameter sort="true" />
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+2007060120070601009Johnydari ",EXCHANGE,20070601,20070601,00:00:04,00:00:04,1,3,
Feed,"Server1",User,"romydario@yahoo.es",Account_Code,"Johnydari",2,EXEMSNT,1,EXBYSNT,366438

"CSR+2007060120070601006Joeu79 ",EXCHANGE,20070601,20070601,00:00:04,14:03:05,1,3,Feed,
"Server1",User,"u797991@ntmcon02.emn.com",Account_Code,"Joeu79",2,EXEMRCV,15,EXBYRCV,610456
```

The Account_Code identifier was added to the output CSR using the input identifier User and the account code conversion table.

Values assigned to the `Account_Code` field in the output file are as follows:

`MoveField MOVE0`
: User romydario@yahoo.es is converted to John using the account code conversion table and is appended with `ydari` due to offset 4 and length 5 resulting in `Account_Code = Johnydari`.

`MoveField MOVE1`
: User u797991@ntmcon02.emn.com is converted to Joe using the account code conversion table and is appended with `u79` due to offset 1 and length 3 resulting in `Account_Code = Joeu79`.

To define a literal as a MoveField:

If you specify a literal as the move field, the literal value will be included at the beginning, end, or within the account code as defined in the account code conversion table. For example, if you specify the literal `abc` as a move field, the definition would be as follows:

```
<MoveField name="MOVE0" literal="abc" />
```

Example:

Assume that the Account Code conversion table is defined as follows:

```
romy,,John@0
u797,,Joe@1
```

Assume that the following CSR file is the input:

```
EXCHANGE,20070601,20070601,00:00:04,00:00:04,,2,"Feed","Server1","User","romydario@yahoo.es"
,4,EXEMRCV,0,EXBYRCV,0,EXEMSNT,1,EXBYSNT,366438

EXCHANGE,20070601,20070601,00:00:04,14:03:05,,2,"Feed","Server1","User","u797991@ntmcon02.emn.com"
,4,EXEMRCV,15,EXBYRCV,610456,EXEMSNT,0,EXBYSNT,0
```

If the `MoveField` definition appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true" trace="false" stopOnStageFailure="true" >
 <Identifiers>
  <Identifier name="Account_Code">
   <FromIdentifiers>
    <FromIdentifier name="User" offset="1" length="12"/>
   </FromIdentifiers>
   <MoveFields>
    <MoveField name="MOVE0" literal="User1" />
    <MoveField name="MOVE1" literal="User2" />
   </MoveFields>
  </Identifier>
 </Identifiers>
 <Files>
  <File name="AcctConv.txt" type="table" />
  <File name="MyException.txt" type="exception" format="CSROutput" />
 </Files>
 <Parameters>
  <Parameter exceptionProcess="true" />
  <Parameter sort="true" />
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+2007060120070601009JohnUser1 ",EXCHANGE,20070601,20070601,00:00:04,00:00:04,1,3,Feed,
"Server1",User,"romydario@yahoo.es",Account_Code,"JohnUser1",2,EXEMSNT,1,EXBYSNT,366438

"CSR+2007060120070601008JoeUser2 ",EXCHANGE,20070601,20070601,00:00:04,14:03:05,1,3,Feed,
"Server1",User,"u797991@ntmcon02.emn.com",Account_Code,"JoeUser2",2,EXEMRCV,15,EXBYRCV,610456
```

The Account_Code identifier was added to the output CSR using the input identifier User and the account code conversion table.

Values assigned to the `Account_Code` field in the output file are as follows:

`MoveField MOVE0`
: User romydario@yahoo.es is converted to John using the account code conversion table and is appended with the literal `User1` resulting in `Account_Code = JohnUser1`.

`MoveField MOVE1`
: User u797991@ntmcon02.emn.com is converted to Joe using the account code conversion table and is appended with the literal `User2` resulting in `Account_Code = JoeUser2`.

Defining the account code conversion table:

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

For example:

```
<Files>
   <File name=" AcctTabl.txt  type="table"/>
</Files>
```

If the table is in the process definition directory.

Or

```
<Files>
   <File name="/usr/local/cims/cimslab/AcctTabl.txt " type="table"/>
</Files>
```

If the table is not in the process definition directory.

Related concepts:

"Creating the account code conversion table (Integrator program)" on page 180
This topic describes how to create the account code conversion table.

Enabling exception processing:

Exception processing instructs the system to write all records that do not match an entry in the account code conversion table to an exception file.

To enable exception file processing, include the `exceptionProcess` parameter in the in the job file as follows

```
<Parameters>
   <Parameter exceptionProcess="true"/>
</Parameters>
```

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

You must also specify the name of the exception file in the `Files` element of the job file. For example:

```
<Files>
   <File name=" Exception.txtâ€  type="exception" format="CSRPlusOutput"/>
</Files>
```

Considerations for exception processing

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

Creating the account code conversion table (Integrator program):

The integrator program is capable of reading conversion definitions from both a file and from the database.

Managing conversions in a file

The account code conversion table is an ASCII text file that contains the definitions that are required to convert the identifier values defined by the account code input field to user-defined output account codes.

The following example shows the account code conversion table:

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

The fields in the account code conversion table entries are low identifier, high identifier, and the account code to which you want to convert (maximum of 127 characters).

Considerations when creating account code conversion table files

Consider the following when creating the account code conversion table:

- The account code conversion table is case-sensitive. For convenience, you can enter uppercase values in the table and use the parameter `upperCase="true"`. This is especially helpful it you are using one account code conversion table for multiple process definitions. This ensures that account code input field values that are lowercase or mixed case are processed.
- Each record in the table can be up to 385 characters (129 for the low identifier, 129 for the high identifier, and 127 for the new account code).
- The identifier fields follow these rules:
  - Each low and high identifier field is compared to the corresponding account code input field values. If the compares are true, the account code is assigned.
  - The low identifier fields are padded with x'00' and the high value fields are padded with x'FF'.
  - The high identifier field is set equal to the low field plus the high padding when the high identifier value is null.
  - You can enter up to 10 levels of 127 characters each for the low and high identifier fields. Each level must be delimited by a colon (:).

For example, if you type AA as the low identifier and CC as the high identifier, all account codes that begin with characters greater than or equal to AA or less than or equal to CC are converted to the specified account code. If you type AAAAAAAA:BB as the low identifier and CCCCCCCC:DD as the high identifier, the Account Code lookup for the first level will match characters greater than or equal to AAAAAAAA or less than or equal to CCCCCCCC for the first identifier and the Account Code lookup for the second level will match characters greater than or equal to BB or less than or equal to DD for the second identifier. This process is repeated for each level/identifier specified.

The asterisk (*) and the question mark (?) characters can be used as wildcard characters for both the low and high identifiers. Using the wildcard characters can degrade performance; use them only when needed.

Note: Although the maximum number of characters for each field level is 127, the overall length of all levels added together cannot exceed 127 characters.

- You can include a default account code as the last entry in the account code conversion table by leaving the low and high identifier fields empty (e.g., ",, DEFAULTCODE"). In this case, all records that contain identifier values that do not match an entry in the account code conversion table will be matched to the default code.
- If you do not include a default account code, input file records that contain identifier values that do not match an entry in the account code conversion table are written to the output CSR+ file with the unconverted account code input field value as the account code.

  If you want unmatched records to go to an exception file for reprocessing at a later time, you need to specify the parameter `exceptionProcess="true"`. If you enable exception file processing, make sure that you have not included a default account code entry in the account code conversion table.
- If the parameter `writeNoMatch="true"` is included in the job file, a message is written to the trace file showing the identifier values from the input file that were not matched during account code conversion. If this statement is not present, only the total number of unmatched records is provided. The system prints up to 1000 messages.
- The number of definition entries that you can enter in the table is limited only by the memory available to Integrator.

Account code conversion example (Integrator program):

This topic provides and example of an account code conversion.

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

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

DDDCCCCUUUUUUUU

D = Division (three characters) (highest level)

C = Cost Center (four characters)

U = User (eight characters) (lowest level)

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

Table 52. User identifier values

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

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

Define the account code and conversion parameters in the job file

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

```
<Stage name="IdentifierConversionFromTable" active="true">
   <Identifiers>
     <Identifier name="Account_Code">
       <FromIdentifiers>
          <FromIdentifier name="User" offset="1" length="7"/>
       </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"/>
   </Parameters>
</Stage>
```

Create the account code conversion table

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

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

Setting up conversion mappings

Use this topic to maintain conversion mapping definitions. Conversion mappings are used as an input into the Account Code Conversion feature in the Integrator program. Mappings defined here can be exported as an Account Code Conversion table for use in the Integrator program.

Adding conversion mappings

You can add or edit conversion mapping information using the `UpdateConversionFromRecord` stage. Refer to the Reference section for more details.

About exception file processing

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

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

Setting up shifts

In Administration Console, you can set rates for a rate code by shift. This optional feature enables you to set different rates based on the time of day.

You can use either of the following methods to determine the shift codes that are used for processing. These shift codes appear in output CSR+ records produced by the Acct program..

- Use the shift code from the input file records. If a shift code is not included in the records, the default shift code is 1.
- If you do not want to use the shift codes in the input record for processing, you can recalculate the shifts using the start date/time in the records and the Acct control statement parameter `SHIFT`.

  Note: The shift recalculation feature is not available for the Integrator program.

Note: Regardless of whether you use the rate codes from the input file or recalculate the rate codes, you need to include the Bill program control statement parameter `USE SHIFT CODES` in the job file. If this control statement is not present, the Bill program uses the default rate shift 1 regardless of the shift code that appears in the CSR+ file.

Using the SHIFT control statement

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

The Acct program compares the start dates and times in the input records to the daily shifts that you specify using the `SHIFT` control statement parameters and

calculates new shift codes. The recalculated shift codes rather than the original shift codes appear in the Acct output CSR+ file.

The syntax for the SHIFT control statement is as follows:

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

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

`code` = 1–9 (the shift code)

`end time` = the end time of the shift

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

Transferring files

Data processing is performed on the SmartCloud Cost Management application server. However, many of the files that contain the data are on separate systems. When SmartCloud Cost Management cannot access the data on other systems, you must transfer the files to the application server during the data collection process. You can use several methods to transfer the files.

SmartCloud Cost Management Data Collectors collect data from the following sources:

- Usage metering data generated by other applications. This data is usually contained in standard usage metering files, such as log files, but might also be contained in other files, such as trace files or reports. For example the WebSphere data collectors collect log files; the data collector for SAP collects report files.
- Usage metering files generated by SmartCloud Cost Management. For example, the Windows Processor data collector gathers usage data for processes running on Windows operating systems and produces a log file of the data.

Remote files can be accessed by the SmartCloud Cost Management application server using shared paths (Windows) or mounted drives (UNIX or Linux). On computers that are running the Microsoft Windows operating system, shared paths include mapped (net used) drives and Universal Naming Convention (UNC) drives.

Note: If you are accessing a UNC file path, the SmartCloud Cost Management application server must be given access to the UNC location before you run the file transfer step in the job file. The application server will not map (net use) the UNC drive automatically.

In some situations, the SmartCloud Cost Management application server cannot access remote files, and you must transfer the files from the remote system to the application server. Following are two scenarios in which the application server cannot access remote files:

- The SmartCloud Cost Management application server is not set up to automatically access the remote system through shared paths or mounted drives. For example, you might want to transfer log files created by the Windows Processor data collector from a Microsoft Windows operating system server to the SmartCloud Cost Management application server on UNIX or Linux. In these types of situations, you must transfer files from the remote system to the local SmartCloud Cost Management application server system.

- The data collector cannot be configured to automatically push files from the source system to the SmartCloud Cost Management application server. For example, the application server is behind a firewall.

How files are transferred

You can initiate a file transfer in either or the following ways:

- Pull-based file transfer is initiated from the SmartCloud Cost Management application server. Pulling files from a remote system to the application server can be implemented using a FileTransfer step within a job file. To transfer files using a job file you can:
  - Modify the data collector job file to pull the remote files to the SmartCloud Cost Management application server.
  - Run a separate job file to pull the remote files to the SmartCloud Cost Management application server.
- Push-based file transfer is initiated from the remote system. Pushing files from remote systems to the SmartCloud Cost Management application server is implemented outside of SmartCloud Cost Management using file transfer tools such as FTP. Consult your network administrator for assistance.

The file transfer method that you should use depends on your particular environment and operating system. The file transfer should occur before the job file for the data collector is run.

Sample file transfer job files

The following sample job files provide different file transfer scenarios and assume that the "from" system is remote to the SmartCloud Cost Management application server, and the "to" system is the application server. These sample job files “pull” remote files to the SmartCloud Cost Management application server. The sample job files are in the <*SCCM_install_dir*>/samples/jobfiles directory.

- `SampleFileTransferFTP_withPW.xml` This job file contains file transfer examples that show you how to pull files from remote directories to the SmartCloud Cost Management application server using the File Transfer Protocol (FTP).
- `SampleFileTransferSSH_withPW.xml` This job file contains file transfer examples that show you how to pull files from remote drives and directories to the SmartCloud Cost Management application server using the Secure Shell (SSH) file transfer protocol. In general, you will use this method of file transfer if you are transferring files from a UNIX or Linux system to theSmartCloud Cost Management application server on UNIX or Linux or if you are transferring files from a Windows system (that uses Cygwin's SSH support) to the SmartCloud Cost Management application server on UNIX or Linux.
- `SampleFileTransferLocal_noPW.xml` This job file contains file transfer examples that show you how to pull files from drives and directories that are considered to be local to the SmartCloud Cost Management application server. Local drives can include hard disk drives that are connected to the application server, Windows mapped drives, Windows UNC drives, and UNIX mounted drives.

Although the sample job files show how to copy files from a remote system to the SmartCloud Cost Management application server, you can also copy files from the application server to a remote system. For example, you might want to push collectors that are created by your organization to a remote system.

Sample FTP file transfer job file

The *<SCCM_install_dir>*\samples\jobfiles\SampleFileTransferFTP_withPW.xml job file contains file transfer examples that show how to pull files from remote directories to the SmartCloud Cost Management application server using FTP.

The SampleFileTransferFTP_withPW.xml job files provides these examples:

- Transfer of a remote Microsoft Windows, UNIX, or Linux file to a local Microsoft Windows, UNIX, or Linux system using the macro %CollectorLogs% in the directory structure.
- Transfer of a remote Microsoft Windows, UNIX, or Linux file to a local Microsoft Windows system using a Microsoft Windows drive and directory structure.
- Transfer of multiple remote files to a local Microsoft Windows, UNIX, or Linux system using the macro %CollectorLogs% in the directory structure.
- Transfer of multiple remote files to a local Microsoft Windows system using a Microsoft Windows drive and directory structure.

Tips for using the sample job file

Consider the following when you are working with the sample job file. For additional information about the file, refer to the comment information in the file.

- You must specify values for the serverName, userId, and userPassword parameters. You can use either a clear text password or an encrypted password in the userPassword parameter value. To encrypt the password, use the SmartCloud Cost Management password encryption feature.
- The transferType parameter is optional. The values are binary (default) or ascii.
- The from and to parameter values can include SmartCloud Cost Managementmacros, such as %CollectorLogs% and %LogDate_End%.
- The ftp:// file transfer protocol can be used for either the from or the to parameter, depending on whether you are transferring the file from or to the remote server. The from parameter includes the ftp:// protocol, because the sample job file demonstrates how to transfer files to the SmartCloud Cost Management application server.
- The from location can include wildcards in the file name. If the from parameter value includes file name wildcards, the to parameter must specify a directory name (no file names).
- The non-FTP file path in the from or to parameter value can be any of the following:
  - Local path (UNIX, Linux, or Microsoft Windows)
  - UNC path (Microsoft Windows)
  - Mounted drive (UNIX or Linux)

Related tasks:

"Using an encrypted password to connect to a remote system" on page 190
You can use a job file to connect to a remote system to transfer or install files. The parameters required to connect to the remote system include a password parameter. You can use either a clear text password or an encrypted password in the password parameter value.

Sample SSH file transfer job file

The <*SCCM_install_dir*>\samples\jobfiles\SampleFileTransferSSH_withPW.xml job file contains file transfer examples that show how to pull files from remote drives and directories to the SmartCloud Cost Management application server using the Secure Shell (SSH) file transfer protocol. In general, use this file transfer method if you are transferring files from a UNIX or Linux system to the SmartCloud Cost Management application server on UNIX or Linux, or if you are transferring files from a Microsoft Windows system (that uses Cygwin's SSH support) to the SmartCloud Cost Management application server on UNIX or Linux.

The SampleFileTransferSSH_withPW.xml job files provides these examples:

- Transfer of a remote file from a UNIX or Linux SSH system to a local UNIX, Linux, or Microsoft Windows system using the macro %CollectorLogs% in the directory structure.
- Transfer of a remote file from a UNIX or Linux SSH system to a local Microsoft Windows system using a Microsoft Windows drive and directory structure.
- Transfer of a remote file from a Microsoft Windows (Cygwin) SSH system to a local UNIX, Linux, or Microsoft Windows system using the macro %CollectorLogs% in the directory structure.
- Transfer of a remote file from a Microsoft Windows (Cygwin) SSH system to a local Microsoft Windows system using a Microsoft Windows drive and directory structure.
- Transfer of multiple remote files to a local UNIX, Linux, or Microsoft Windows system using the macro %CollectorLogs% in the directory structure.

Tips for using the sample job file

Consider the following when you are working with the SampleFileTransferSSH_withPW.xml job file. For additional information about the file, refer to the comment information in the file.

- You must specify values for the serverName, userId, and userPassword parameters. You can use either a clear text password or an encrypted password in the userPassword parameter value. To encrypt the password, use the SmartCloud Cost Management password encryption feature.
- If you are using a keyfile to authenticate with the ssh service (Cygwin on Windows) or the ssh daemon (sshd on UNIX or Linux), the following are applicable:
  - The fully qualified name of the private key for the SmartCloud Cost Management application server must be specified in the KeyFileName parameter.
  - If you created the keyfile using a pass phrase, the userPassword parameter must contain the current valid pass phrase. If you created the keyfile using an empty pass phrase, you must specify any text string in theuserPassword parameter. The userPassword parameter cannot be blank.
  - The user ID on the remote system must be configured to trust the SmartCloud Cost Management application server user ID. This means that if the keyfile path is specified, the public key for the SmartCloud Cost Management application server user ID must be in the authorized keys file of the remote user.
- The ssh:// file transfer protocol can be used for either the from or the to parameter, depending on whether you are transferring the file from or to the

remote server. The `from` parameter includes the `ssh://` protocol, because the sample job file demonstrates how to transfer files to the SmartCloud Cost Management application server.

- The `from` and `to` parameters must use the UNIX-style naming convention, which uses forward slashes (for example, `ssh:///anInstalledCollectorDir/CollectorData.txt`), regardless of whether you are transferring files from a UNIX or Linux system or a Microsoft Windows system.
- You can use the ssh file transfer protocol on UNIX, Linux, or Microsoft Windows systems. However, before you run a job file that uses the ssh protocol, verify that you can successfully issue ssh commands from the SmartCloud Cost Management application server to the remote ssh system.
- The `from` and `to` parameter values can include SmartCloud Cost Management macros, such as `%CollectorLogs%` and `%LogDate_End%`.
- The `from` location can include wildcards in the file name. If the `from` parameter value includes file name wildcards, the `to` parameter must specify a directory name (no file names).

Using Cygwin SSH support on Windows

When you transfer Windows files using Cygwin ssh support, you must use the Cygwin-configured "mount prefix". For example, the Cygwin default mount prefix is `/cygdrive/<drive>/`, which is the equivalent to the forward slash (`/`) context on UNIX or Linux systems. For example, `C:\` on Windows is `/cygdrive/c` for Cygwin.

The following path examples show the notation for accessing the Window-equivalent files on a Cygwin target system. The target system is specified in the `from` parameter value. Note that Cygwin path names are case sensitive and use the forward slash path separator:

- `/cygdrive/c/Program Files c:/Program Files`
- `/cygdrive/k/docs1 k:/docs1`

If you specify a path without a drive letter for a Cygwin target, the location of the directory is relative to the top-level Cygwin directory. For example, if you specify `/bin` as the path, and Cygwin is installed in the `/cygwin` directory, the location on the Cygwin target is assumed to be `/cygwin/bin`.

Related tasks:

"Using an encrypted password to connect to a remote system" on page 190
You can use a job file to connect to a remote system to transfer or install files. The parameters required to connect to the remote system include a password parameter. You can use either a clear text password or an encrypted password in the password parameter value.

Sample local file transfer job file

The `<SCCM_install_dir>\samples\jobfiles\SampleFileTransferLocal_noPW.xml` job file contains file transfer examples that show how to pull files from drives and directories that are considered to be local to the SmartCloud Cost Management application server. Local drives can include hard disk drives that are connected to the application server, Microsoft Windows mapped drives, Microsoft Windows UNC drives, and UNIX mounted drives.

Note: Local file transfer uses operating system security and does not require that you to provide authentication credentials within a job file. Local file transfer is the preferred method for moving, copying, and deleting files on the local operating system.

The `SampleFileTransferLocal_noPW.xml` job files provides these examples:

- Transfer of a local Microsoft Windows file to a local Microsoft Windows system. This example transfers a file from one local directory to another on the SmartCloud Cost Management application server. Both directories are on the same hard disk drive.
- Transfer of a Microsoft Windows file on a mapped drive to a Microsoft Windows local system. This example transfers a file from a mapped drive to a local directory on the SmartCloud Cost Management application server. The source file defined in the `from` parameter is on a mapped drive with a valid drive letter.
- Transfer of a Microsoft Windows file on UNC drive to Microsoft Windows local system. This example transfers a file from a UNC location to a local directory on the SmartCloud Cost Management application server.
- Transfer of multiple Microsoft Windows files (on a mapped drive) to a Microsoft Windows local system. This example transfers multiple files from one mapped drive and directory to a local directory on the SmartCloud Cost Management application server.
- Transfer of a local UNIX or Linux file to a local UNIX or Linux system. This example transfers a file from one local directory to another on the SmartCloud Cost Management application server.
- Transfer of a UNIX or Linux file on a mounted drive to a local UNIX or Linux system. This example transfers a file from a mounted drive to a local directory on the SmartCloud Cost Management application server.
- Transfer of multiple UNIX or Linux files on a mounted drive to a local UNIX or Linux system. This example transfers multiple files from a mounted drive to a local directory on the SmartCloud Cost Management application server.

Considerations when using the sample job file

The following are key considerations when you are working with the `SampleFileTransferLocal_noPW.xml` job file. For additional information about the file, refer to the comment information in the file.

- The `file://` file transfer protocol is be used for both the `from` and the `to` parameters.
- The `from` location can be a local path, a Windows UNC path, a Windows mapped drive, or a UNIX or Linux mounted drive path
- The `to` location must be a path that is local to the `from` location. On Windows, this can include a local hard disk drive, a UNC path or a mapped drive. On UNIX or Linux, this can include a local directory or a UNIX mounted drive path
- The `from` and `to` parameter values can include SmartCloud Cost Management macros such as `%CollectorLogs%` and `%LogDate_End%`.
- The `from` location can include wildcards in the file name. If the `from` parameter value includes file name wildcards, the `to` parameter must specify a directory name (no file names).

Using an encrypted password to connect to a remote system

You can use a job file to connect to a remote system to transfer or install files. The parameters required to connect to the remote system include a password parameter. You can use either a clear text password or an encrypted password in the password parameter value.

About this task

To encrypt the password, use the `passwordManager` command, as shown in the following example:

```
passwordManager -e <mypassword>
```

Where `<mypassword>` is the password used to connect to the remote server. This command returns an encrypted password that you can enter in the password parameter value.

The following shows the usage for the `passwordManager` command.

```
passwordManager [-h | -help | ?] | [-e | -encrypt <clear password>]

               -h | -help | ? : display usage
               -e | -encrypt <clear password> : password in clear text
```

Related reference:

"Sample FTP file transfer job file" on page 186
The *<SCCM_install_dir>*`\samples\jobfiles\SampleFileTransferFTP_withPW.xml` job file contains file transfer examples that show how to pull files from remote directories to the SmartCloud Cost Management application server using FTP.

"Sample SSH file transfer job file" on page 187
The *<SCCM_install_dir>*`\samples\jobfiles\SampleFileTransferSSH_withPW.xml` job file contains file transfer examples that show how to pull files from remote drives and directories to the SmartCloud Cost Management application server using the Secure Shell (SSH) file transfer protocol. In general, use this file transfer method if you are transferring files from a UNIX or Linux system to the SmartCloud Cost Management application server on UNIX or Linux, or if you are transferring files from a Microsoft Windows system (that uses Cygwin's SSH support) to the SmartCloud Cost Management application server on UNIX or Linux.

Sample SMB file transfer job file
The *<SCCM_install_dir>*`\samples\jobfiles\SampleFileTransferWindows_withPW.xml` job file contains file transfer examples that show how to pull the files from remote Windows drives and directories to the SmartCloud Cost Management application server. The file transfer uses the Windows SMB file transfer protocol. This type of file transfer requires a server name, user ID, and password for the remote system.

File encoding

This topic describes the default input and output file encoding for SmartCloud Cost Management. It also discusses how the encoding of input files can be overridden. The encoding for output files cannot be overridden.

Default file encoding

This topic describes the default input and output file encoding for the SmartCloud Cost Management.

Integrator input and output file encoding

The following are the default encoding schemes for the files used by the Integrator program:

- Input files that are specified by the `Input` element are opened, by default, in system encoding. This is because SmartCloud Cost Management does not control the building of collector logs.
- Files specified by the `CSRInput` attribute are opened by default in encoding auto-detection mode. The auto-detection mode will automatically detect the encoding of a file when it is opened for reading.
- A file specified by the `CreateIdentifierFromTable` attribute of the `Stage` element is opened in system encoding.
- A file specified by the `IdentifierConversionFromTable` attribute of the `Stage` element is opened in system encoding.
- The files specified by the `CSROutput` and `CSRPlusOutput` attributes of the `Stage` element are written in UTF-8.

Job Runner input file encoding

The following are the default encoding schemes for the files used by the Job Runner program.

- The Scan program opens all input CSR files in encoding auto-detection mode. This can be overridden.

Processing Engine input and output file encoding

The following are the default encoding schemes for the files used by SmartCloud Cost Management Processing Engine.

- The SmartCloud Cost Management Processing Engine writes all output CSR and CSR+ and database load files (Ident, Summary, Detail) in UTF-8.
- The account code conversion table in used by the Acct program is opened in system encoding.

Related reference:

"Overriding default file encoding"
This topic describes how to override these defaults.

Overriding default file encoding

This topic describes how you can override the default encoding of input files.

Overriding default file encoding for the integrator

To override the default encoding, use the `encoding` attribute in the job file as shown in the following sections.

You can override the default system encoding for files specified by the Integrator using the `encoding` attribute of the`Input` element.

- A CSRInput file can be processed in UTF-16BE encoding using the following code:

```
<Input name="CSRInput" active="true">
 <Files>
  <File name="%ProcessFolder%/CurrentCSR.txt" encoding="UTF-16BE"/>
 </Files>
</Input>
```

- A CSRInput file can be processed in system encoding using the following code:

```
<Input name="CSRInput" active="true">
 <Files>
  <File name="%ProcessFolder%/CurrentCSR.txt" encoding="system"/>
 </Files>
</Input>
```

Overriding default file encoding for the Scan program

The following code segment illustrates how to override the default system encoding for files that are opened by the Scan program:

```
<Step   id="Scan"
 description="Scan MSExchange"
 type="Process"
 programName="Scan"
  programType="java"
 active="true">
  <Parameters>
  <Parameter retainFileDate="false"/>
   <Parameter allowMissingFiles="false"/>
  <Parameter allowEmptyFiles="false"/>
  <Parameter useStepFiles="false"/>
  <Parameter encoding="system"/>
 </Parameters>
</Step>
```

Overriding default file encoding for identifier creation and conversion tables

The following code segment illustrates how to override the default system encoding for identifier creation and conversion tables:

```
<File name="Accttabl.txt" type="table" encoding="UTF-8"/>
```

Overriding default file encoding for the Process Engine

The following code segment illustrates how to override the default system encoding for input file that is processed by the SmartCloud Cost Management:

```
<Step   id="Process"
 description="Standard Processing for MSExchange"
  type="Process"
 programName="Acct"
 programType="java"
  active="true">
 <Acct>
 <Parameters>
  <Parameter inputFileEncoding="system"/>
  </Parameters>
  </Acct>
</Step>
```

Chapter 5. Administering reports

Use these tasks to set up, create, and view reports.

Working with Cognos based Tivoli Common Reporting

The following topics are intended for use with IBM Cognos 10.2 Business Intelligence. IBM Cognos 10.2 is a Web product with integrated reporting, analysis, scorecarding, and event management features.

About the Tivoli Common Reporting application

The Cognos based Tivoli Common Reporting application provides comprehensive cost accounting, chargeback, and resource reporting in an easy-to-use, browser-based environment.

Cognos based Tivoli Common Reporting includes the following features that ensure that users receive the data they need in a clear format:

- **Drill down -** Cognos based Tivoli Common Reporting invoices and many other reports include drill down that enables users to view detailed cost and usage information.
- **Ad-hoc reporting -** Cognos based Tivoli Common Reporting allows you to create ad-hoc reports simply and quickly using the Cognos Query Studio tool. This tool allows you to access the information you require quickly.
- **Simplified report creation -** Cognos based Tivoli Common Reporting requires no knowledge of the database or SQL to create reports. All data is available to you to ensure you can find the information you need quickly. Reports can be published to a public or private area for use.

About SmartCloud Cost Management reports

SmartCloud Cost Management includes standard reports for Cognos based Tivoli Common Reporting. Cognos based Tivoli Common Reporting provides access to Cognos Business Intelligence, a market leading Business Intelligence tool. Reports are written using the Query Studio and Report Studio Cognos tools depending on whether you are creating an ad-hoc or a more professional report. Report definitions are stored within Cognos and they can be exported as XML or as a ZIP export file if required. You can use the standard reports provided as templates to create custom reports for your organization. Reports created specifically for your organization are referred to as “custom reports” in this guide to differentiate them from the SmartCloud Cost Management standard reports.

Date Ranges

The reports can be run for various date ranges including both Calendar and Fiscal Calendar ranges. The following table describes the date ranges and in which reporting tool they are available.

© Copyright IBM Corp. 2013, 2015

Table 53. Date Ranges

Date Option	Cognos based Tivoli Common Reporting	Description
All	Yes	All dates
Date Range Below / Custom	Yes	A user defined date range
Today	Yes	Today
Yesterday	Yes	Yesterday
Last 7 days	Yes	Previous 7 days including current date
Last 30 days	Yes	Previous 30 days including current date
Last 90 days	Yes	Previous 90 days including current date
Last 365 days	Yes	Previous 365 days including current date
Current Week to date	Yes	The start of the calendar week to the current date
Current Month to date	Yes	The start of the calendar month to the current date
Current Year to date	Yes	The start of the calendar year to the current date
Last Week	Yes	The previous calendar week
Last Month	Yes	The previous calendar month
Last year	Yes	The previous calendar year
Current Week	Yes	The current calendar week
Current Month	Yes	The current calendar month
Current Year	Yes	The current calendar year
Current Period	Yes	The current fiscal period as defined in the `CIMSCalendar` table
Previous Period	Yes	The previous fiscal period as defined in the `CIMSCalendar` table
Fiscal Year	Yes	The fiscal year as defined in the `CIMSCalendar` table
Previous Fiscal Year	Yes	The previous fiscal year as defined in the `CIMSCalendar` table
Fiscal Year to date	Yes	The start of the fiscal year as defined in the `CIMSCalendar` table to the current date
Previous Fiscal Year to date	Yes	The previous fiscal year as defined in the `CIMSCalendar` table to the current date of the previous year

The Reports and Spreadsheets support a fiscal year with either 12 or 13 periods.

Note: The start of a calendar week is Sunday.

Running reports

The SmartCloud Cost Management 2.1.0.5 reports are run using the Common Reporting Interface. This section describes how to find and run a report.

Before you begin

Before using SmartCloud Cost Management reports:

- Tivoli Common Reporting 3.1.0.1 must be installed.
- A SmartCloud Cost Management schema data source must be set up within the Common Reporting option.
- The SmartCloud Cost Management 2.1.0.5 Cognos package must have been imported. For more information about this, see the *Configuring after installation guide*.

Procedure

1. Navigate to the Tivoli Common Reporting interface. See related links for more information about logging into the reporting interface based on the Tivoli Common Reporting version you are using.
2. The SmartCloud Cost Management reports are visible as **IBM SmartCloud Cost Management (SCCM)**.
3. Click the **IBM SmartCloud Cost Management (SCCM)** package in the **Public Folders** tab.
4. Click the folder where the report you want to run is located. A list of reports are shown.
5. Click the report name to run the report. Alternatively you can run a report by clicking the run with menu options icon.
6. Specify the parameters required (if any) to limit the data returned.
7. Click the **Finish** button.

Results

The report has been generated.

Related reference:

For Tivoli Common Reporting 3.1.0.1 - Logging in to the reporting interface
For Tivoli Common Reporting 3.1.0.1 - Logging in to the reporting interface

Using the Cognos toolbar

Once a report has been run, there are various options available to you on the toolbar. This topic describes some of these options.

Note: The following toolbar tips can be used when using Cognos functionality:

- To return to Cognos Connection after a report has been run, click the return Menu icon.

  Note: Do not use the browser **Back** button to return to Cognos Connection. This is browser-related and is independent of Cognos.
- To rerun a report, click the run menu icon.
- To view the report in a different format, for example, PDF, click the view menu icon and select the format you want to view the report in.

More information about using the toolbar is found in the IBM Cognos Connection User Guide 10.2.

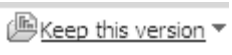
Printing reports

It may be convenient for you to have a printed copy of a report. You may need to review a report when your computer is not available, or you may need to take a copy of a report to a meeting.

About this task

More information about printing reports is found in the Reports and Cubes section of the IBM Cognos 8 Administration Guide 10.2.

Saving reports

After a report has run, you can save the report with the data using the **Keep this Version** menu 

icon. This icon allows you to email the report, save the report as a report version or save the report as a report view.

About this task

If you save the report as a report version, the report is saved in Cognos and is accessed using the report version icon for the report in **Cognos Connection**. If you save the report as a report view then you see another version of the report with a report version as before in **Cognos Connection**.

If you save the report to your desktop, use the view in  menu icon to view the report in PDF or Excel format and save the report from there.

For more information about saving reports, see the IBM Cognos Connection User Guide 10.2.

Report properties

In **Cognos Connection**, a properties  icon is available for each report. This allows you to set default properties for a report. For example, properties such as description, the format of the report, the number of report versions to keep, default prompt values and what action to take when the report name is clicked on in the User Interface.

For more information about report properties, see the *IBM Cognos Connection User Guide 10.2*.

Creating Dashboard reports

IBM SmartCloud Cost Management contains reports that can be used to demonstrate two different methods for implementing dashboard reports.

1. The first method uses two reports that are used to build a page. This page allows the user to arrange individual reports on the page and use Cognos portlets for additional functionality. These reports are:
   - **Top N Account Charges Pie Chart** – this report shows the percentage distribution of charges for the top N accounts with the highest charges. This report uses a pie chart to display the data.
   - **Top N Account Charges** – this report shows a list of the top N accounts with the highest charges. In this report, the data is displayed as a list.

2. The second method uses a single report that contains several reporting objects, such as graphs and lists arranged as a dashboard but in a single manageable report. This method uses the following report:
   - **Top N Rate Group and Rate Resource Usage** – this report shows a pie chart and tables for both the top N rate groups and rate codes with the highest % increase in charges between the previous two periods.

Note: For more information about Pages and Dashboards see the *IBM Cognos Connection user guide* at http://www.ibm.com/support/knowledgecenter/SSEP7J_10.2.0/com.ibm.swg.ba.cognos.ug_cc.10.2.0.doc/t_managing_pages.html.

The following topics describe how to create the dashboard reports using these two methods.

Creating a sample dashboard report

Use this task to create the sample dashboard for the **Top N Account Charges Pie Chart** and **Top N Account Charges** reports along with a Cognos Navigator object for running other reports from the dashboard.

Before you begin

Before proceeding to create the sample dashboard report, you must ensure that the following prerequisites are completed:

- The **Top N Account Charges Pie Chart** and **Top N Account Charges** reports displayed on the dashboard must have been run and saved previously. This is required so you do not have to wait for the report to run before the page is displayed. Therefore, it is advisable to schedule the reports using the **Save the report** option, so that they can be displayed immediately on the dashboard.
- For demonstration purposes, a single saved report output can be created using the **Run with options** button for the report. Alternatively, if these reports must show current data, then they can be scheduled to run regularly.

Procedure

1. In IBM Cognos Connection, click the new page button  .
2. Type the name, and select a location for your page and click **Next**.
3. In the **Set columns and layout** page, set the number of columns to 2, and the column widths to 60% on the left column and 40% on the right column.
4. Click **Add...** for the first column.
5. In the **Available entries** box, click **Cognos Content**.
6. Click **Cognos Viewer** and click the add  menu icon to add it to the page and click **OK**.
7. Click **Cognos Navigator** and click the add  menu icon to add it to the page and then click **OK**.
8. Click **Add...** for the second column.
9. In the **Available entries** box, click **Cognos Content**.
10. Click the **Cognos Viewer** and click the add  menu icon to add it to the page and click **OK**.
11. Click **Next**,
12. If required add a title and click **Next**.
13. Check the **View the page** option and click **Finish**.

14. The page is displayed with the empty portlets for the reports and a navigator for accessing the reports.
15. Click the edit  icon in the upper left portlet.
16. In the entry section, click **Select an entry....**
17. Go to the **Dashboard Reports / Individual Dashboard Reports** folder in the **IBM SmartCloud Cost Management (SCCM)** package and select the Top N Account Charges and click **OK**.
18. In the view options, specify a height of 350 pixels and click **OK**.
19. Click the edit  icon on the lower left portlet.
20. In the folder section, click **Select a Folder...**
21. Go to the **IBM SmartCloud Cost Management (SCCM)** package and click **OK** and then click **OK** again.
22. Click the edit  icon on the right side portlet.
23. In the **Entry** section, click **Select an entry....**
24. Go to the **Dashboard Reports / Individual Dashboard Reports** folder in the **IBM SmartCloud Cost Management (SCCM)** package and select the **Top N Account Charges Pie Chart** and click **OK**.
25. In the view options, specify a height of 500 pixels and click **OK**.
26. Click **Edit this page**  icon for the page.
27. Select the **Page Style** tab and click the **Hide title bars** check box and click **OK**.

Results

The sample dashboard is now created.

Creating Page dashboard reports

The second method uses a single report that contains several reporting objects, such as graphs and lists arranged as a dashboard but in a single manageable report. The Page dashboard uses a single report to create the dashboard report.

Before you begin

Before proceeding to create the sample dashboard report using method one, you must ensure that the following is completed:

- The **Top N Rate Group and Rate Resource Usage** report displayed on the dashboard must have been run and saved previously. This is required so you do not have to wait for the report to run before the page is displayed. Therefore, it is advisable to schedule the reports using the **Save the report** option, so that they can be displayed immediately on the dashboard.
- For demonstration purposes, a single saved report output can be created using the **Run with options** button for the report. Alternatively, if these reports need to show current data, then they can be scheduled to run regularly.

Procedure

1. In IBM Cognos Connection, click the new page button  .
2. Type the name, and select a location for your page and click **Next**.
3. In the **Set columns and layout** page, click **Add...**.

4. In the **Available entries** box, click **Dashboard**.
5. Click the **Multi-page** check box and click the add menu icon to the page and click **OK**.
6. Click **Next**.
7. Click the **View the page** check box option and click **Finish**.
8. Click the edit icon in the right-side of the portlet.
9. In the entry section, click **Select an entry....**
10. Go to the **Dashboard Reports / Individual Dashboard Reports** folder in the **IBM SmartCloud Cost Management (SCCM)** package and select the **Page Dasboard Reports** folder and click **OK**.
11. Click **OK**.

Results

The dashboard is now created. Additional reports can be added to the **Page Dashboard Reports** folder and these will then be included on the page. These pages can be added to the set of tab pages as follows:

1. Click the tab menu icon and select the **Add tabs...** option.
2. Go to the folder where the dashboard was saved in (see step 2) and select the dashboard.
3. Click **OK**.
4. The dashboard is then added into the list of available tabs .

If you want one of these reports to be your home page when using Common reporting, use the **Set View as Home** option under the home icon .

Working with custom reports

Use this section to learn more about working with custom reports in SmartCloud Cost Management.

Creating, editing, and saving custom reports

This section provides the steps for creating, editing, and saving custom reports using Report Studio in the SmartCloud Cost Management Cognos Reporting application.

Creating or updating a custom report:

Custom reports are created using the IBM Cognos 10.2 Query Studio and Report Studio tools. Query Studio is used for ad hoc reporting and provides a basic level of report formatting for presentation. Report Studio is the tool used to create the standard reports and provides improved querying of the database and report formatting functionality.

About this task

Report Studio provides a Web-based tool that professional report authors use to build sophisticated, multiple-page, multiple-query reports against multiple databases. You can create any reports that your company requires, such as invoices, statements, and weekly sales and inventory reports.

Note: By default, access to Report Studio is only granted to the Administration user. If you need to use Report Studio, you must contact your System Administrator to grant access to this capability.

The following procedure details how to access Report Studio:

Procedure

1. Log in to the reporting interface. Based on the version of Tivoli Common Reporting you are running, refer to the appropriate related topic.
2. In the **Common Reporting** window, click **Report Studio** from the **Launch** drop-down list. This opens up the **Report Studio**, a Web-based application.
3. Use the menu controls to create a new report or edit existing ones by formatting the layout and manipulating the data that appears in the report.
4. Save your report, and run it anytime you need to present on its underlying data.

What to do next

For more information about Web-based report authoring, see *Report Studio User Guide* available by clicking **F1** from the **Report Studio**.

Related reference:

For Tivoli Common Reporting 3.1.0.1 - Logging in to the reporting interface
For Tivoli Common Reporting 3.1.0.1 - Logging in to the reporting interface

Saving a custom report:

You can share a report with others by saving the report in a location that is accessible to other users, such as in the public folders. Public folders typically contain reports that are of interest to many users. A Report definition is saved in Report Studio by using the save option in the File menu (**File** > **Save**) or by clicking on the save icon. If you are saving for the first time, you are prompted for a directory in Cognos to save the report. If the report is a private report then save the report to a personal folder otherwise the report can be saved to the Public folders area in order for other users to use the report.

About this task

For more information about saving reports, see the *Report Studio User Guide 10.2*.

Using report parameters

Cognos based Tivoli Common Reporting provides various methods for adding parameters to reports. Parameters in Cognos are based on prompts whereby a prompt is created that has an associated parameter value. This parameter value can be used in the filter in a query or it can be displayed on a report page.

Prompt controls provide the user interface in which the questions are asked. Parameter values provide the answers to the questions.

Various prompts are provided that allow the user to enter the relevant parameter values. Radio buttons, drop down lists, select and other search prompts are available for use. Some prompt types allow the report writer to provide a static list of values to the user. Prompts assist the user when filtering the data shown in the report when the report is run.

For more information about prompts, see the section on *Adding Prompts to Filter Data* in the Report Studio User Guide 10.2

Simple parameters:

The following task topics explain how to create a prompt (parameter) and add it to the main query of a report to filter the returned data.

Filter a query using a parameter:

When a prompt (parameter) is added to the main query of a report, Cognos automatically prompts for a value when the report is run.

Before you begin

A report must be created consisting of a query using the `[Summary (SCCM Consolidation)].[Summary]` query subject and the report page shows the items from the query. The report must also be open in Report studio.

About this task

This task uses an example to filter a query, to return summary data for a specific rate code

Procedure

1. Navigate to the query in the **Query Explorer**.
2. Drag the `[Rates (SCCM Consolidation)].[Rates].[Rate Code]` query item into the Detail Filters window in the **Query Explorer**.
3. `Append = ?RateCode?` to the rate code so that the filter looks like this: `[Rates (SCCM Consolidation)].[Rates].[Rate Code] = ?RateCode?`.
4. Run the report by clicking the run icon.

Results

When the report is run, a prompt page is displayed asking the user to enter a value for the rate code and the query is filtered using the value entered.

Example

Create a prompt page:

This topic describes how to build a prompt page for an item that is already displayed on the Report page by using the **Build Prompt Page** button.

Before you begin

A report has been created consisting of the following:

- A query using the `[Summary (SCCM Consolidation)].[Summary]` Query subject.
- The report page showing the items from the query which include the `[Summary (SCCM Consolidation)].[Summary].[Start Date]` query item.

The report must also be open in Report studio.

Procedure

1. Navigate to the report page in the report and click the `[Summary (SCCM Consolidation)].[Summary].[Start Date]` item displayed.
2. Click the **Build Prompt Page** button to create a prompt page with a suitable prompt type to use.
3. Run the report by clicking the run icon.

Results

When the report is run, you are prompted to enter two dates. Only data between the two dates will be shown. In the **Page Explorer** in the report, a prompt page is created and in the query, a filter has been added:

```
[Summary (SCCM Consolidation)].[Summary].[Start Date] in_range ?Start Date?
```

This filter restricts the data returned to be within the range you selected in the prompt page. The **Build Prompt Page** button works with any query item on the report page and can be used as the starting point for creating customized prompt pages.

Create a prompt page manually:

This topics describes how to manually create a prompt page that provides a high level of flexibility for setting parameter values.

About this task

A prompt page is created by adding a prompt page in the **Page Explorer** . The Cognos prompt tools are used to add the relevant prompt types to the report page after which the relevant properties are set. Filters using the parameters created are added to the relevant queries. This method allows you to use a prompt that specifies the values that a user selects when running the report, either from a static list of values or a query.

For more information about creating a prompt page manually, see the *Adding Prompts to Filter Data* section of the Report Studio User Guide 10.2

Existing parameters:

In the Cognos Model for SmartCloud Cost Management, some parameters are defined and are used for some of the common filters required in the standard reports.

The existing parameters are described as follows:

`AccountStructureIndex`

The **`AccountStructureIndex`** parameter is used to restrict the data returned for the account code level within an account code structure. A calculated item is added to the model to index the account structures available to users with the default for the report group having an index of 0. When selecting the account code structure to use in the report, the user selects the account code structure name but the parameter is assigned the index number for that account code structure.

The index number is used by the Account Structure Index Filter in the model to restrict the Account Code Structure Levels returned in the Account Level prompt.

AccountCodeLevel

The **AccountCodeLevel** parameter is used for the selected level within the account code structure. It is used to determine which account codes are displayed in the reports and the account code prompts.

This is used by the Account Level Prompt Filter to restrict the account codes shown in the account code prompts. In addition, it is also used to populate two hidden prompts that hold the account code level offset (start position) and length as defined in the account code structure.

StartingAccountCodeType and StartingAccountCode

The **StartingAccountCodeType** and **StartingAccountCode** prompts represent the values entered when you select the account code lower boundary used to run the report. A calculation is defined in the model to convert the values entered for these prompts into a single item and corresponding filters, Start Summary Account Code Filter and Start Detail Account Code Filter have been created to restrict the values returned from the summary and detail data respectively.

EndingAccountCodeType and EndingAccountCode

The **EndingAccountCodeType** and **EndingAccountCode** parameters hold the account code level offset (start position) and length as defined in the account code structure for the chosen account code structure level. These parameters are populated when an account code structure level is selected within hidden prompts. The parameters are used by the main query in the reports to return the correct account codes.

DateFilter

The **DateFilter** parameter holds an integer value that represents the date option selected on the prompt page. This is used with the Accounting Calendar Prompt Filter and Resource Usage Calendar Prompt Filter to restrict the data retrieved from the summary and detail tables for the relevant dates with the first filter using the Accounting Date and the second using the Resource Usage dates to filter the data.

The following shows the values and the dates represented:

- 0 - All
- 1 - Date Range Below
- 2 - Today
- 3 - Yesterday
- 4 - Last 7 days
- 5 - Last 30 days
- 6 - Last 90 days
- 7 - Last 365 days
- 8 - Current Week to date
- 9 - Current Month to date
- 10 - Current Year to date
- 11 - Last Week
- 12 - Last Month
- 13 - Last year
- 14 - Current Week
- 15 - Current Month
- 16 - Current Year

- 17 - Current Period
- 18 - Previous Period
- 19 - Fiscal Year
- 20 - Previous Fiscal Year
- 21 - Fiscal Year to date
- 22 - Previous Fiscal Year to date

Rate Group
The **Rate Group** parameter holds the rate group value selected on the prompt page and is used to restrict the data to the rate group selected. It is used in conjunction with the Rate Group Prompt Filter.

RateCode
The **RateCode** parameter holds the `ratecode` value selected on the prompt page and is used to restrict the data to the rate code selected. It is used in conjunction with the Rate Code Prompt Filter.

Identifier
The **Identifier** parameter holds the identifier number of the selected identifier in the identifier prompt. It is used to restrict the data to the identifier selected. It is used in the Identifier Number Prompt Filter.

Note: The **Identifier** parameter is not used in reports that require more than one identifier prompt.

Shift The **Shift** parameter holds the value of the shift from the summary data when drilling down to the detail data. This parameter is used to ensure that the resource usage shift code is between the relevant boundaries. This is used with the Resource Usage Shift Code Prompt Filter.

RatePattern
The **RatePattern** parameter holds the Rate Pattern value(s) selected on the prompt page and is used to restrict the data to the rate pattern selected. It is used in conjunction with the Rate Pattern Prompt.

RatePatternFilter
The **RatePatternFilter** parameter holds a token that contains the rate patterns selected on the prompt page and is used to restrict the data to the rate pattern group selected. It is used in conjunction with the Rate Pattern Group Prompt Filter.

RateTable
The **RateTable** parameter holds the Rate Table value selected on the prompt page and is used to restrict the data to the rate table selected. It is used in conjunction with the Rate Table Prompt Filter.

Using report filters

You can use a filter to specify the subset of records that the report retrieves. Any data that does not meet the criteria is eliminated from the report, which can improve performance. A number of filters are predefined for use in the Standard reports. This section describes those filters and explains how they restrict the data returned.

Table 54. Predefined filters

Filter	Description
Account Code Prompt Content	Used to filter the account codes returned for the account code prompts, so that their length does not exceed the length defined in the Configuration options or account code level selected.
Account Structure Index	Used to filter the data returned to show a selected account code structure. For more information about this filter, see the `AccountStructureIndex` section of the *Existing parameters* topic.
Account Level Prompt	Used to filter the account code data returned for a selected level within the account code structure. For more information about this filter, see the `AccountCodeLevel` section of the *Existing parameters* topic.
Start Detail Account Code Filter	Used to filter the detail data returned to show account codes with values greater than the `accountcode` selected. For more information about this filter, see the `StartingAccountCodeType` and `StartingAccountCode` section of the *Existing parameters* topic.
End Detail Account Code Filter	Used to filter the detail data to show account codes with values less than the `accountcode` selected. For more information about this filter, see the `EndingAccountCodeType` and `EndingAccountCode` section of the *Existing parameters* topic.
Start Summary Account Code Filter	Used to filter the summary data returned to show the account codes with values greater than the `accountcode` selected. For more information about this filter, see the `StartingAccountCodeType` and `StartingAccountCode` section of the *Existing parameters* topic.
End Summary Account Code Filter	Used to filter the summary data returned to show the account codes with values less than the `accountcode` selected. For more information about this filter, see the `EndingAccountCodeType` and `EndingAccountCode` section of the *Existing parameters* topic.
Calendar Year Prompt Content	Used to filter the data shown in the calendar prompts to show those years less than the (current year + 1).
Accounting Calendar Prompt	Used to filter the summary and detail data to show the dates for the selected date option using the accounting calendar dates.
Resource Usage Calendar Prompt	Used to filter the detail data to show the dates for the selected date option using the resource usage calendar dates.

Table 54. Predefined filters (continued)

Filter	Description
Identifier Number Prompt	Used to filter the detail data for a specified identifier number.
Rate Group Prompt	Used to filter the summary or detail data to return records for a specified rate group.
Rate Code Prompt	Used to filter the summary or detail data to return records for a specified rate code.
Start Budget Account Code	Used to filter the client budget data returned to show the account codes with values greater than the **accountcode** selected. For more information about this filter, see the **StartingAccountCodeType** and **StartingAccountCode** section of the Existing parameters topic.
End Budget Account Code	Used to filter the client budget data returned to show the account codes with values less than the **accountcode** selected. For more information about this filter, see the **EndingAccountCodeType** and **EndingAccountCode** section of the Existing parameters topic.
Drilldown Rate Prompt	Used to filter the rate retrieved from the detail table when drilling down to the detail level.
Rate Pattern Prompt	Used to filter the summary or detail data to return records for a specified rate pattern.
Rate Pattern Group Prompt Filter	Used to filter the summary or detail data to return records for a specified rate pattern type (Billable / All).
Rate Table Prompt Filter	Used to filter the summary or detail data to return records for a specified rate table.

For information on filtering data, you can refer to the Adding Prompts to Filter Data section of the Report Studio User Guide 10.2

Template reports

Three template reports are provided in the Template folder as part of the SmartCloud Cost Management package. This section describes these reports and how they are used.

The three template reports are:

- Template
- Template Account Code Date
- Template Account Code Year

Template report

This report is used to hold the shared header and footer used by all of the standard reports. More details on this can be found in the *Rebranding reports* section.

The following two concept topics describe the Template Account Code Date and the Template Account Code Year template reports.

Template Account Code Date:

This report is used as a template for custom reports. As some of the prompt functionality is complex to implement, this report has been created with the basic functionality added.

The report contains the JavaScript functionality described in the following sections:

- Date prompts
- Setting the date to the current Date
- Account code prompts
- Cascading prompts on a single page
- For more information about this JavaScript functionality, see these topics for more detailed information.

A Custom Report is created using this report by doing the following:

1. Open the Template Account Code Date Report from the Template folder in Report Studio.
2. Save the report to another folder with a different name.
3. In Report Studio, add the query items you want to view on the report to the 'Main Query'.
4. On the Report page, place the items from the 'Main Query' query in the list object.

The report can now be run and returns the relevant data restricted by the Date and Account Codes selected.

These reports are designed to filter the data retrieved from the Summary data. If you want to apply these reports to the detail data, complete the following steps:

1. Open the Template Account Code Date Report from the Template folder in Report Studio.
2. Save the report to another folder with a different name.
3. In the 'Main Query' query, remove the following filters from the Detail Filters:

   ```
   [Prompts (SCCM Consolidation)].[Start Summary Account Code Filter]
   [Prompts (SCCM Consolidation)].[End Summary Account Code Filter]
   ```

4. Add the following filters from the model to the Detail Filters:

   ```
   [Prompts (SCCM Consolidation)].[Start Detail Account Code Filter]
   [Prompts (SCCM Consolidation)].[End Detail Account Code Filter]
   ```

The report is now ready to be amended to show the relevant data.

Template Account Code Year:

This report is used as a template for custom reports. This report is created with the basic functionality added, as the prompt functionality can be complicated to implement.

The report contains the JavaScript functionality described in the following sections:

- Setting the year to the current year
- Account code prompts
- Cascading prompts on a single page

A Custom Report is created using this report by doing the following:

1. Open the Template Account Code Year Report from the Template folder in Report Studio.
2. Save the report to another folder with a different name.
3. In Report Studio, add the query items you want to view on the report to the 'Main Query'.
4. On the Report page, place the items from the 'Main Query' query in the list object.

The report can now be run and returns the relevant data restricted by the Year and Account Codes selected.

These reports are designed to filter the data retrieved from the Summary data. If you want to apply these reports to the detail data, complete the following steps:

1. Open the Template Account Code Year Report from the Template folder in Report Studio.
2. Save the report to another folder with a different name.
3. In the 'Main Query' query, remove the following filters from the Detail Filters:

   ```
   [Prompts (SCCM Consolidation)].[Start Summary Account Code Filter]
   [Prompts (SCCM Consolidation)].[End Summary Account Code Filter]
   ```

4. Add the following filters from the model to the Detail Filters:

   ```
   [Prompts (SCCM Consolidation)].[Start Detail Account Code Filter]
   [Prompts (SCCM Consolidation)].[End Detail Account Code Filter]
   ```

The report is now ready to be amended to show the relevant data.

Rebranding reports

This section describes how to rebrand the standard reports to reflect your company needs.

About this task

The standard reports contain the same header and footer which is defined within the Template report in the Template folder. Each standard report contains a pointer, known as a layout component reference to the header and footer in the Template report. This pointer means that any updates made to the report are reflected in all of the Standard Reports.

The following procedure describes how to modify the headers or footers in all Standard Reports:

Procedure

1. Open the Template Report from the Template folder in Report Studio.
2. Make the relevant amendments to the header or footer in the report.

   Note: The header constitutes all objects within the `RTSubtitleBlock1` block and the footer constitutes all objects within the `FooterTable` table.
3. Save the report.

Results

When a standard report is run or opened with Report Studio, the new header or footer is displayed in the report.

For more details about layout component references, see the *Reuse a Layout Object* topic in the Report Studio Professional Authoring User Guide 10.2

Working with the SmartCloud Cost Management Cognos model

The source files for the SmartCloud Cost Management 2.1.0.5 Cognos model are supplied with the product. This section describes how these source files can be customized to allow you to extend the model to accommodate any custom needs that you have.

The SmartCloud Cost Management Cognos model is customized using the IBM Cognos Framework Manager tool.

Note: This is not supplied as part of Tivoli Common Reporting and requires a separate license and installation. Use the Jazz for Service Management knowledge center to find information about Tivoli Common Reporting 3.1.0.1.

The model can be customized and made visible to users. This method ensures that custom changes will persist after new releases of the SmartCloud Cost Management Cognos model are made.

For more information about the IBM Cognos Framework Manager, see the Cognos Framework Manager User Guide 10.2.

Accessing the Cognos model:

This topic describes how to access the Cognos model.

About this task

Once Framework Manager is installed, the model is accessed by doing the following:

Procedure

Start Framework Manager and open the `SCCM.cpf` file.

- Tivoli Common Reporting 3.1.0.1 – `<SCCM_install_dir>\setup\console\reportscognos\model`

Results

You can now access the Cognos model.

Adding custom objects to the Cognos model:

This section describes how to add custom objects into the Cognos model without affecting future releases of the SmartCloud Cost Management Cognos model.

About this task

Custom objects should be added to a branched version of the project and then merged back into any future releases of the SmartCloud Cost Management Cognos model when they are released. All changes must be made in the Custom namespaces. This means that any additional objects or amended versions of objects must be created under the Custom (SmartCloud Cost Management database) or the Custom (SmartCloud Cost Management Consolidation) namespaces. Changes

made to any other parts of the model may invalidate support for the SmartCloud Cost Management Cognos reports.

Procedure

1. Start Framework Manager and open the `SCCM.cpf` file.
   - Tivoli Common Reporting 3.1.0.1 – `<SCCM_install_dir>\setup\console\reportscognos\model`
2. In Framework Manager open the model and navigate to the **Project** menu and select **Branch to...**
3. On the menu, enter a new name and a new location for the project and click **OK**.
4. Close the SmartCloud Cost Management project.
5. Open the new project created in step 3.

Results

Changes can now be made in the Custom namespaces of the project created in step 3.

Publishing the custom model:

This section describes how the custom changes to the model are published so that the report users can see them. The model is published to the Cognos server as a package that you can use.

Procedure

1. Once all changes have been made, navigate to the Package section, located at the end of the project viewer in Framework Manager.
2. Click the **Click to Edit** link in the definition property for the package or alternatively, right-click on the package name and select **Edit**.
3. Ensure that the relevant namespaces and query subjects have a green tick against them and click **OK**.

   Note: By Default, the Custom (SCCM Database) and Custom (SCCM Consolidation) namespaces are not published.
4. Right-click on the IBM SmartCloud Cost Management package and select **Publish Packages...** and a menu is displayed.
5. Navigate through the menu options as required and publish the package.

   Note: There are 2 packages defined in the model. The **IBM SmartCloud Cost Management (SCCM)** package is the latest model and should be used for all reports. The **IBM Tivoli Usage and Accounting Manager (TUAM)** package is also available for legacy reasons.

Results

The changes are now visible. If you have Report Studio or Query Studio already open, you must refresh the package before the changes are seen. For more information about publishing packages, see the *Publishing Packages* topic in the Framework Manager User Guide 8.4.1.

Additional resources

Use the following resources to find more information about IBM Tivoli Common Reporting and IBM Cognos 10.2 Business Intelligence.

Click the **?** link on the **Common Reporting** portlet. This allows you to view a Cognos Quick Tour, Getting Started information, and additional IBM Cognos 10.2 Business Intelligence documentation. In addition, you can review the following content:

- Monitor the SmartCloud Cost Management wiki for the Common Reporting best practices: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/IBM%20SmartCloud%20Cost%20Management/page/Common%20Reporting
- IBM developerWorks Community Space for Tivoli Common Reporting: https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=9caf63c9-15a1-4a03-96b3-8fc700f3a364. This is a clearing house of Tivoli Common Reporting information, guidelines, "how to" articles, and forums.
- Use the Jazz for Service Management knowledge center at http://www.ibm.com/support/knowledgecenter/SSEKCU_1.1.0.1/com.ibm.psc.doc_1.1.0.1/psc_ic-homepage.html to find information about Tivoli Common Reporting 3.1.0.1.
- Service Management Connect (SMC) is an Integrated Service Management technical community on developerWorks, which connects developers and product experts with clients and partners through blogs, forums, and wikis. Check out SMC for reporting best practices and other useful information. To access SMC, join the Register and create a developerWorks profile: https://www.ibm.com/developerworks/dwwi/jsp/Register.jsp

In addition, web resources are also available to help you get started with Tivoli Common Reporting:

- *The Cognos Proven Practices documentation*: http://www.ibm.com/developerworks/data/library/cognos/cognosprovenpractices.html - Created by Cognos experts from real-life customer experiences, Cognos Proven Practices is your source for rich technical information that is tried, tested, and proven to help you succeed with Cognos products in your specific technology environment.

Chapter 6. Administering the database

Use these tasks to track database loads, administer database objects, work with database tables, and upgrade the database.

Related reference:

Schema updates for the 2.1.0.5 release
The product schema was updated as part of the SmartCloud Cost Management 2.1.0.5 release.

Tracking database loads

You can track the history of Summary, Detail, Ident, and optional Resource files that have been loaded to or deleted from the database.

About this task

To track database loads complete the following steps:

Procedure

1. In Administration Console, click **Tasks** > **Load Tracking** .
2. On the Load Tracking page, use the following:

   Select Feed
   Select the appropriate feed if it is not displayed. This shows the folder that contains the file that was processed to create the feed.

   Delete Feed
   Delete all loads that are associated with the currently selected feed.

   Refresh
   Click to refresh the information on the page. The page shows any updates to the data.

   Accounting Dates
   Select to view all loads that are ordered by their accounting date for the selected feed. You can drill down as follows:
   - Expand the year node to view the period or periods that are associated with that year.
   - Expand the period node to view the start date or dates that are associated with that period.
   - Expand the start date node to view load or loads that are associated with that start date.
   - Expand the load node to view accounting codes, details, or ident information.

   Loaded Dates
   Select to view all loads that are ordered by their loaded date for the selected feed. You can drill down as follows:
   - Expand the year node to view the month or months that are associated with that year.
   - Expand the month node to view the loaded date or dates that are associated with that month.

© Copyright IBM Corp. 2013, 2015

- Expand the loaded date node to view the load or loads that are associated with that loaded date.
- Expand the loads node to view the accounting codes, details, or ident information.

Deleting loads

When the **Accounting Dates** option is selected you can delete loads that are associated with the accounting year, period, start date or you can delete individual loads. To do this:

- Right click the accounting year, period, start date, or the individual load node and select **Delete Loads**.

When the **Loaded Dates** option is selected you can delete loads that are associated with the accounting year, month, loaded day or you can delete individual loads. To do this:

- Right click the accounting year, month, loaded day, or the individual load node and select **Delete Loads**.

Click **Yes** to delete the load for the selected option or **No** to retain the entry. If you remove the load, the load entry is removed from the tree menu.

Note: If you delete all loads from a feed, the feed dropdown tree menu is refreshed and the feed is removed.

Administering database objects

All database object creation scripts are stored as XML files. The standard database objects that are provided with SmartCloud Cost Management are in the `<SCCM_install_dir>/setup/dbobjects/standard` directory.

Any custom objects that are created for you organization must be stored in the `<SCCM_install_dir>/setup/dbobjects/custom` directory.

Note: If an object of the same name is in both the standard and custom directory, SmartCloud Cost Management uses the object in the custom directory.

Note: Certain reports are dependent on database objects. Dropping or altering database objects could cause the reports to fail.

Note: A DDL script can be generated using the `DataAccessManager` command-line utility containing the SQL for creating the database objects.

SmartCloud Cost Management includes the following types of database objects:

- **Tables.** tables are database objects that contain all the data in a database. A table definition is a collection of columns. In tables, data is organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field within the record.
- **Views**. Each view is essentially a saved SQL statement that performs a query that allows the system to generate a report or reports.
- **Stored Procedures**. Each procedure is set of SQL statements that can perform both queries and actions that allow the system to generate a report or reports. Stored procedures can accept parameters at run time, whereas views cannot.
- **Indexes.** A table index allows the database program to find data in a table without scanning the entire table. There can be multiple indexes per table or no index at all.

- **Triggers.** A trigger is a special type of stored procedure that executes when data in a specified table is modified. A trigger can query other tables and can include complex SQL statements.

Administering database tables

The `DataAccessManager` command-line utility can be used to export data from the database tables or load into the database tables.

Note: All database objects, including tables, are XML files that can be customized for your site.

About Exporting and Loading Tables

The following is a description of the drop, create, export, and load functions used for database tables:

- *Exporting* a table from the database creates a comma-delimited text file with the same name as the table, but does not remove the table or its data. Exporting a table is useful as a means of backup, as the file can be reloaded into the database if required.
- *Loading* a table into the database first deletes all data currently in the table and then loads the table from the file in the directory that you specify. You can load files that were previously exported from the database or you can load files from another source such as a mainframe file.

  The file must have the same name as the table, for example, `SCRate.txt` for the `SCRate` table and it must contain the following:

  – The table column headings in the first record of the file.
  – All required fields for the table provided in comma-delimited format. To determine whether a field is required, view the table properties in the applicable database administration tool. Fields that are not specified as nullable must be included in the file.

    Note: When importing data into database tables, additional tables may also need to loaded. If no data is required to be loaded into these additional tables, a file containing no data must be created for each additional table. This file can be generated using the Export Table utility.

    Restriction: To avoid a constraint violation error, when loading data into the database tables, the tables must be loaded in a particular sequence. For more information about this, see the *Loading table dependencies* related topic.

    For an example of the required format for a file, you can export the table to a file.

    CAUTION:
    Extreme caution should be used when loading a table. Deleting or overwriting data can cause unexpected results. Be sure a backup exists before performing this procedure.

Loading table dependencies

When loading data into the database tables , any data referenced in other tables should also be loaded to avoid a constraint violation. A constraint violation occurs when data is loaded into the database tables that references data that does not exist.

The following table shows what tables, if any, should also be loaded when loading the original tables.

Note: This is only a guide and it is up to the user to ensure that all relevant data has been loaded.

Table 55. Load table dependencies

Dependent table	Related tables
CIMSACCOUNTCODECONVERSION	CIMSPROCESSDEFINITION
	CIMSPROCDEFINITIONMAPPING
CIMSCLIENT	CIMSCLIENTBUDGET
	CIMSCLIENTCONTACT
	CIMSCLIENTCONTACTNUMBER
CIMSCLIENTBUDGET	CIMSCLIENT
CIMSCLIENTCONTACT	CIMSCLIENT
	CIMSCLIENTCONTACTNUMBER
CIMSCLIENTCONTACTNUMBER	CIMSCLIENT
CIMSIDENT	CIMSDETAILIDENT
CIMSDETAILIDENT	CIMSIDENT
CIMSPROCDEFINITIONMAPPING	CIMSPROCESSDEFINITION
	CIMSACCOUNTCODECONVERSION
CIMSPROCESSDEFINITION	CIMSPROCESSDEFINITION
	CIMSPROCDEFINITIONMAPPING
	CIMSACCOUNTCODECONVERSION
	CIMSPRORATESOURCE
	CIMSPRORATETARGET
CIMSPRORATESOURCE	CIMSPROCESSDEFINITION
	CIMSPRORATETARGET
CIMSPRORATETARGET	CIMSPROCESSDEFINITION
	CIMSPRORATESOURCE
CIMSRATEGROUP	SCRATE
	SCUNITS
	SCRATETABLE
	SCRATESHIFT

Table 55. Load table dependencies (continued)

Dependent table	Related tables
CIMSREPORT	CIMSREPORTCUSTOMFIELDS
	CIMSREPORTDISTRIBUTION
	CIMSREPORTDISTRIBUTIONCYCLE
	CIMSREPORTDISTRIBUTIONPARM
	CIMSREPORTTOREPORTGROUP
	CIMSREPORTGROUP
	CIMSREPORTSTART
	CIMSREPORTOTREPORTRATEGROUP
CIMSREPORTCUSTOMFIELDS	CIMSREPORT
CIMSREPORTDISTRIBUTION	CIMSREPORT
	CIMSREPORTDISTRIBUTIONCYCLE
	CIMSREPORTDISTRIBUTIONPARM
CIMSREPORTDISTRIBUTIONCYCLE	CIMSREPORTDISTRIBUTION
	CIMSREPORTDISTRIBUTIONPARM
CIMSREPORTDISTRIBUTIONPARM	CIMSREPORT
	CIMSREPORTDISTRIBUTIONCYCLE
CIMSREPORTGROUP	CIMSREPORTTOREPORTGROUP
	CIMSREPORTSTART
	CIMSREPORT
	CIMSREPORTDISTRIBUTION
CIMSREPORTSTART	CIMSREPORTGROUP
	CIMSREPORTTOREPORTGROUP
	CIMSREPORT
	CIMSREPORTDISTRIBUTION
CIMSREPORTTOREPORTGROUP	CIMSREPORTGROUP
	CIMSREPORTSTART
	CIMSREPORT
	CIMSREPORTDISTRIBUTION
CIMSUSER	CIMSUSERTOUSERGROUP
	CIMSUSERGROUP
CIMSUSERCONFIGOPTIONS	CIMSUSER
CIMSUSERGROUP	CIMSUSERTOUSERGROUP
	CIMSUSER
CIMSUSERGROUPCONFIGOPTIONS	CIMSUSERGROUP
SCRATE	CIMSRATEGROUP
	SCRATETABLE
	SCUNITS
	SCRATESHIFT

Table 55. Load table dependencies (continued)

Dependent table	Related tables
SCRATESHIFT	SCRATE
	SCUNITS
	SCRATETABLE
SCRATETABLE	SCRATE
	SCUNITS
	SCRATESHIFT
SCUNITS	SCRATE

Related tasks:

Managing database tables
All database object creation scripts are stored as XML files and can be customized for your site. The standard database tables that are provided with SmartCloud Cost Management are in the <*SCCM_install_dir*>\setup\dbobjects\standard directory. Any custom objects that are created for your organization must be stored in the <SCCM_install_dir>\setup\dbobjects\custom directory. If a table of the same name is in both the standard and custom directory, SmartCloud Cost Management uses the table in the custom directory.

DataAccessManager command-line utility

DataAccessManager provides several management utilities for the data layer. These utilities include extracting DDL statements of all SmartCloud Cost Management database objects, populating tables in the SmartCloud Cost Management database with their initial values, and setting the database version number.

DataAccessManager is invoked from a command line by issuing the command DataAccessManager.sh (Linux).

Show parameters

A list of all available parameters for DataAccessManager can be obtained from the command line by typing DataAccessManager.sh without any parameters, or by specifying the -help parameter.

The output of this utility is similar to the following:

```
Usage: DataAccessManager
                <-ddlextract>
                <-datasource>
                <-outputpath>
                <-dbtype>

Usage: DataAccessManager
                <-populatetables>
                <-datasource>

Usage: DataAccessManager
                <-setdatabaseversion>
                <-version>
                <-datasource>

Usage: DataAccessManager
                <-importtable>
                <-datasource>
                <-tablename>
```

```
                <-inputpath>

Usage: DataAccessManager
                <-exporttable>
                <-datasource>
                <-tablename>
                <-outputpath>
```

DDL extraction

The Data Definition Language (DDL) extraction utility is used to extract all of the Table, Index, Stored Procedure, View and Trigger DDL statements that are part of the SmartCloud Cost Management product into text files.

The output of the DDL Extraction utility is five plain-text files. There is one file for each type of database object. Each file contains all of the DDL statements for its particular database object type. For example, `SCCM_DDL_Table.txt` contains all of the DDL statements for creating tables. The files generated are:

- `SCCM_DDL_Index.txt`
- `SCCM_DDL_StoredProcedure.txt`
- `SCCM_DDL_Table.txt`
- `SCCM_DDL_Trigger.txt`
- `SCCM_DDL_View.txt`

Running DDL extract

DDL Extraction is run by specifying the `-ddlextract` parameter. Several additional options are available for `-ddlextract`. They are:

- `-outputpath <directory path specification>`
- `-dbtype <database type>`
- `-datasource <data source name>`

The `-outputpath` parameter is required. It determines where the generated text files will be placed.

The `-dbtype` parameter is required. It determines the DDL statements which will be extracted based on the type of database. For example, by specifying `DB2LUW`, all DDL statements for the DB2LUW database will be extracted, and DDL statements for other database types will be not be written. The available database types are:

- DB2LUW - IBM DB2 for Linux/Windows

The `-datasource` parameter is optional. If it is specified, macros within the DDL statements are expanded based on information from the specified data source. For example, the database object prefix macro (`%DBOBJECTPREFIX%`) is replaced with actual value from the specified data source definition. If this parameter is not specified, the macros in the DDL statements will not be expanded, but will be retained.

Two examples of running the DDL Extraction utility and the sample console output are below.

This example shows how to run DDL Extract and expand the macros with information from the data source.

```
DataAccessManager -ddlextract -outputpath /tmp -dbtype DB2LUW -datasource SCO_DB2

56 Tables
```

```
15 Indexes
69 Stored procedures
0 Triggers
13 Views
```

This example shows how to run the same command without macro expansion.

```
dataaccessmanager -ddlextract -outputpath /tmp -dbtype DB2LUW

56 Tables
15 Indexes
69 Stored procedures
0 Triggers
13 Views
```

Populate tables

The Populate Tables utility is used to load an initial data set into the SmartCloud Cost Management database tables. The data loaded is the same as when the SmartCloud Cost Management database is first initialized.

This utility can be used in several scenarios, including populating SmartCloud Cost Management tables when these tables have been created outside of the SmartCloud Cost Management UI.

Note: This utility will remove and overwrite data in several key SmartCloud Cost Management tables, such as Client and Rate.

Populate Tables is run by specifying the `-populatetables` parameter. A single, required option is available for `-populatetables`:

```
-datasource <data source name>
```

The `-datasource` parameter is used to specify which SmartCloud Cost Management database will have existing table data removed and then populated with an initial set of values.

An example of running the Populate Tables utility and the resulting output are included below.

```
DataAccessManager.sh -populatetables -datasource MYDB
Loading configuration table.
Loaded configuration table.
Loading tables...
Loaded 19 tables.
```

Setting the database version

The Set Database Version utility is used to set the version of the SmartCloud Cost Management database. The most common use of this utility is to set the current database version to be lower than it truly is in order to allow the repeat of a database upgrade.

Before you begin

The SmartCloud Cost Management Administrator's UI will not allow an upgrade to run more than once if the database is at, or greater than, the latest upgrade available. The database version can be set to a lower number to allow the SmartCloud Cost Management UI to run a database upgrade again. For example, the database version can be set from `002.003` to `002.002` to allow the 002.003 database upgrade to be run.

About this task

The format of the database version number is `mmm.nnn`, where `mmm` is a 3-digit number identifying the major version of the database, and `nnn` is a 3-digit number identifying the minor version of the database. For example, 002.003. Set Database Version is run by specifying the `-setdatabaseversion` parameter. Two required options are available for `-setdatabaseversion`:

- `-datasource <data source name>`
- `-version <version number>`

The `-datasource` parameter is used to specify which SmartCloud Cost Management database will have its version number modified. The `-version` parameter is used to specify the new version number of the database.

Example

An example of running the Set Database Version utility and resulting output is below. In this example, the database version number of the database pointed to by the MYDB data source will be updated to version number 002.003. The output from the command shows the original version number and the upgraded version number.

```
DataAccessManager.sh -setdatabaseversion -datasource MYDB -version 002.003
002.002-->002.003
```

Importing data into a database table

The Import Table utility is used to load data from a comma delimited file directly into the database table in SmartCloud Cost Management. This utility can be used when there are large amounts of data that must be loaded or if the data is already held in comma delimited files. This avoids inserting records one at a time through the User Interface.

Loading a table into the database deletes all the data currently in the table and then loads the table from the file in the directory that you specify. You can load files that were previously exported from the database or you can load files from another source such as a mainframe file.

Restriction: To avoid a constraint violation error, when loading data into the database tables, the tables must be loaded in a particular sequence. For more information about this, see the Loading table dependencies related topic.

The file must have the same name as the table, for example, `SCRate.txt` for the `SCRate` table and it must contain the following:

- The table column headings in the first record of the file.
- All required fields for the table must be provided in a comma-delimited format. To determine whether a field is required, view the table properties in the applicable database administration tool. Fields that are not specified as nullable must be included in the file.

When importing data into database tables, additional tables may also need to be loaded. If no data is required to be loaded into these additional tables, a file containing no data must be created for each additional table. This file can be generated using the Export Table utility.

For example, if you want to load the SCRate table, the SCRateShift table must also be loaded, so a file is required for the SCRate table and a second file is required for the SCRateShift table. If the SCRateShift table already has loaded data, then this file can be exported using the Export Table utility. If this table has no data loaded and does not need to have data loaded, a file with no data for the SCRateShift table can be generated using the Export Table utility.

CAUTION:
Extreme caution must be used when loading a table. Deleting or overwriting data can cause unexpected results. Be sure a backup exists before performing this procedure.

It is advisable to backup the data already in the table before importing the new data, as the existing data will be overwritten as part of this process. The export into a database table utility can be used for this if required.

Example - running the Import Table utility and resulting output

In this example, the SCRate and SCRateShift tables are loaded. Two comma delimited files, SCRate.txt and SCRateShift.txt contain the data that must be loaded. These files are stored in the /home/smadmin directory.

```
DataAccessManager.sh -importtable -datasource MYDB -tablename
SCRate,SCRateShift -inputpath /home/smadmin
Loading tables...
Loaded 2 tables.
```

Exporting data into a database table

The Export Table utility is used to export data from a database table in SmartCloud Cost Management to a comma delimited file. This utility can be used when the data needs to be backed up or to extract the data so it can be used with the Import table utility for bulk updates. This is required to avoid having to update records one at a time through the User Interface.

Exporting a table from the database creates a comma-delimited text file with the same name as the table, but it does not remove the table or its data.

Example - Running the Export Table utility and resulting output

In this example, the CIMSClient and CIMSClientContact tables are exported. Two comma delimited files, CIMSClient.txt and CIMSClientContact.txt contain the data that must be created in the /home/smadmin directory containing the data from the database tables.

```
DataAccessManager.sh -exporttable -datasource MYDB -tablename
CIMSClient,CIMSClientContact -outputpath /home/smadmin
Exporting tables...
Exported 2 tables.
```

Chapter 7. Administering data collectors

Use these topics to learn more about data collectors.

Universal Collector overview

The Universal Collector within Integrator was designed to extend the input stage of the integrator and simplify the creation of a new collector.

Note: Usage of the Universal Collector outside of IBM SmartCloud Cost Management Enterprise is subject to certain license restrictions. For more details about these restrictions, see the license terms in the `SCCM_install_dir/license` folder.

An Integrator input stage represents the input processing portion of the collector. The functionality of the collector is to read and process the input and pass each input record onto the next integrator stage. To accomplish this, the collector has to handle all the low-level details which includes opening the input file and exception file, handling the details of reading the input, closing files, creating a database connection, connecting to the database, and other activities. That is where the universal collector becomes useful. It handles all the low-level tasks and allows you to focus on defining the high-level tasks either declaritively by using XML or by programmatically using JAVA, defining the input, defining business processing rules, and defining the output. Currently the framework contains four generic collectors (DATABASE, DELIMITED, FIXEDFIELD and REGEX) along with some product collectors. To create a new collector you must extend from one of the generic or product collectors. If you want to create a new generic collector, then you must extend from the abstract base COLLECTOR.

Creating a new collector using XML

Use this topic to understand how to create a new collector using XML.

About this task

This topic is not in the form of a step by step task. Instead it describes each section of the `CollectorInput` stage as it correlates with the XML data.

Procedure

1. All collectors use the input stage called `CollectorInput`.

```
<Step id="Integrator"
      type="ConvertToCSR"
      programType="java"
      programName="integrator">
  <Integrator>
    <Input name="CollectorInput" active="true">
```

2. The first section of the `CollectorInput` stage is the `Collector` element. This required section defines the name of the collector, and the sub-element attributes define the specific requirements of the collector. The supported names are:
   - `DELIMITED`
   - `DATABASE`
   - `FIXEDFIELD`

© Copyright IBM Corp. 2013, 2015

- REGEX

```
<Collector name="DATABASE|DELIMITED|FIXEDFIELD|REGEX">
</Collector>
```

The following tables describe the applicable elements and attributes by collector name.

Table 56. Elements and attributes specific to DELIMITED

Element	Attribute	Description	Values	Required	Default
RecordDelimiter		Used to define the record delimiter. (Optional: 0 to 1 element)			
	keyword	The character used to delimit the records in the file.	BLANKLINE, FORMFEED, NEWLINE, CUSTOM	Yes	NEWLINE
	customCharacter	A user-defined character used to delimit the records in the file.	Any single valid character.	Yes, if keyword is CUSTOM.	Not applicable.
FieldDelimiter		Used to define the field delimiter. (Optional: 0 to 1 element)			
	keyword	The character used to delimit the fields in a record.	COMMA, TAB, SEMICOLON, COLON, NEWLINE, SPACE, PIPE, BACKSLASH, CUSTOM	Yes	COMMA
	customCharacter	A user-defined character used to delimit the fields in a record.	Any single valid character.	Yes, if keyword is CUSTOM.	Not applicable.
TextFieldQualifier		Used to define the qualifier for fields which contain text. (Optional: 0 to 1 element)			
	keyword	The character used to qualify fields containing text.	DOUBLEQUOTE, SINGLEQUOTE, NONE, CUSTOM	Yes	DOUBLEQUOTE
	customCharacter	A user-defined character used to qualify fields containing text.	Any single valid character.	Yes, if keyword is CUSTOM.	Not applicable.
	escapeMode	The character used to escape the text field qualifier character.	DOUBLED, BACKSLASH	No	DOUBLED
HeaderRecord		Used to define how many records to skip at the beginning of the file. (Optional: 0 to 1 element)			
	skipRecords	The number of records to skip at the beginning of the file.	A positive integer.	Yes	0

Table 56. Elements and attributes specific to DELIMITED (continued)

Element	Attribute	Description	Values	Required	Default
CommentLine		Used to define the comment lines to skip. (Optional: 0 to 1 element)			
	character	The character used to signify a comment line.	Any single valid character.	Yes	By default, comment line processing is off.

DELIMITED Example

```
<Collector name="DELIMITED">
  <RecordDelimiter keyword="FORMFEED"/>
  <FieldDelimiter keyword="TAB"/>
  <TextFieldQualifier keyword="CUSTOM" escapeMode="BACKSLASH" customCharacter="~"/>
  <HeaderRecord skipRecords="3"/>
  <CommentLine character="#"/>
</Collector>
```

Table 57. Elements and attributes specific to FIXEDFIELD

Element	Attribute	Description	Values	Required	Default
RecordDelimiter		Used to define the record delimiter. (Optional: 0 to 1 element)			
	keyword	The character used to delimit the records in the file.	FORMFEED, NEWLINE, CUSTOM	Yes	NEWLINE
	customCharacter	A user-defined character used to delimit the records in the file.	Any single valid character.	Yes, if keyword is CUSTOM.	Not applicable.
HeaderRecord		Used to define how many records to skip at the beginning of the file. (Optional: 0 to 1 element)			
	skipRecords	The number of records to skip at the beginning of the file.	A positive integer.	Yes	0
CommentLine		Used to define the comment lines to skip. (Optional: 0 to 1 element)			
	character	The character used to signify a comment line.	Any single valid character.	Yes	By default, comment line processing is off.

FIXEDFIELD Example

```
<Collector name="FIXEDFIELD">
  <RecordDelimiter keyword="NEWLINE"/>
  <HeaderRecord skipRecords="5"/>
  <CommentLine character="#"/>
</Collector>
```

Table 58. Elements and attributes specific to REGEX

Element	Attribute	Description	Values	Required	Default
RecordDelimiter		Used to define the record delimiter. (Optional: 0 to 1 element)			

Table 58. Elements and attributes specific to REGEX (continued)

Element	Attribute	Description	Values	Required	Default
	keyword	The character used to delimit the records in the file.	FORMFEED, NEWLINE, CUSTOM	Yes	NEWLINE
	customCharacter	A user-defined character used to delimit the records in the file.	Any single valid character.	Yes, if keyword is CUSTOM.	Not applicable.
RecordParser		Used to define a regular expression that will be used to parse the record.			
	regularExpression	A regular expression that is used to parse a record into fields.	A valid Java regular expression.	Yes	Not applicable.
HeaderRecord		Used to define how many records to skip at the beginning of the file. (Optional: 0 to 1 element)			
	skipRecords	The number of records to skip at the beginning of the file.	A positive integer.	Yes	0
CommentLine		Used to define the comment lines to skip. (Optional: 0 to 1 element)			
	character	The character used to signify a comment line.	Any single valid character.	Yes	By default, comment line processing is off.

REGEX Example

```
<Collector name="REGEX">
<RecordDelimiter keyword="NEWLINE"/>
  <RecordParser regularExpression="([^\s]+)\s([^\s]+)\s([^\s]+)\s(\[.+\])
  \s(&quot;.+&quot;)\s([^\s]+)\s([^\s]+)"/>
  <HeaderRecord skipRecords="10"/>
  <CommentLine character="%"/>
</Collector>
```

Table 59. Elements and attributes specific to DATABASE

Element	Attribute	Description	Values	Required	Default
dataSourceName		Used to define the Data Source Name. (Required: 1 element only)			
	dataSourceName	A data source name defined on the Data Source List Maintenance page in Administration Console.	A valid data source name.	Yes	Not applicable.
Statement		Used to define the select statement or a stored procedure. (Required: 1 element only)			

Table 59. Elements and attributes specific to DATABASE (continued)

Element	Attribute	Description	Values	Required	Default
	type	The statement type (SQL statement or a stored procedure).	PROCEDURE, SQL	No	SQL
	text	A SQL statement or a stored procedure name. Use the ? character to represent placeholders for the parameters.	A valid SQL statement or stored procedure call.	Yes	Not applicable.
Parameter		Used to define the parameters for the SQL statement or stored procedure. It is not required if the SQL statement or stored procedure is not accepting any parameters. (Optional: 0 or more elements)			
	src	The source section of the statement parameter.	STATIC, PARAMETER, QUERYPARAMETER	No	STATIC
	sqlType	The SQL type of the statement parameter.	STRING, VARCHAR, CHAR, INTEGER, LONG, DOUBLE, FLOAT, DATE, TIME, TIMESTAMP, BOOLEAN	Yes	Not applicable.
	position	The position of the parameter in the SQL statement or stored procedure signature.	A positive integer.	Yes	Not applicable.
	value	A value for the STATIC parameter.	Any value.	Yes, if src is STATIC.	Not applicable.
	srcName	The name of the source parameter or query parameter.	A name which is defined in the Parameters or QueryParameters section.	Yes, if src is not STATIC.	Not applicable.
	format	Used to format parameters with sqlType of DATE, TIME, and TIMESTAMP.	A valid Java date format string.	No	Not applicable.

DATABASE Example 1

```
<Collector name ="DATABASE">
  <Connection dataSourceName="DBCollector"/>
  <Statement text=" SELECT * FROM CIMSTransaction WHERE (ToDate &gt;= ? AND ToDate &lt;= ?)"/>
  <Parameter src="STATIC" position="1" value="1/1/07 " sqlType="DATE" format="M/d/yy"/>
  <Parameter src="STATIC" position="2" value="12/31/07" sqlType="DATE" format="M/d/yy"/>
</Collector>
```

DATABASE Example 2

The following DATABASE example uses the PROCEDURE statement type and shows how to collect using a stored procedure:

```
<Collector name="DATABASE">
    <Connection dataSourceName="default" />
    <Statement type="PROCEDURE" text="{call GET_SUMMARY(?,?,?,?,?,?,?,?)}" />
    <Parameter src="STATIC"    sqlType="CHAR" position="1" value="" />
    <Parameter src="STATIC"    sqlType="CHAR" position="2" value="zzzz" />
    <Parameter src="STATIC"    sqlType="CHAR" position="3" value="1" />
    <Parameter src="STATIC"    sqlType="CHAR" position="4" value="4" />
```

```
    <Parameter src="PARAMETER" sqlType="CHAR" position="5" srcName="StartDate" />
    <Parameter src="PARAMETER" sqlType="CHAR" position="6" srcName="EndDate" />
    <Parameter src="STATIC"    sqlType="CHAR" position="7" value="admin" />
    <Parameter src="STATIC"    sqlType="CHAR" position="8" value="4" />
</Collector>
```

Note: You can only call stored procedures that return a result set. Calling a stored procedure that does not return a result set will result in an error.

3. The next section of the `CollectorInput` stage is the `Parameters` element. The `Parameters` element is optional (0 or 1 element) and can have one or more `Parameter` elements. Each parameter is defined as a sub-element of the `Parameters` XML element.

   The following table describes the applicable elements and attributes for the `Parameters` element.

Table 60. Elements and attributes specific to `Parameters`

Element	Attribute	Description	Values	Required	Default
Parameter		Used to define a value that can be referenced by name in other parts of the collector. (Required: 1 or more elements)			
	name	A unique name to identify the parameter.	Any value.	Yes	Not applicable.
	value	The value of the parameter.	Any value.	Yes	Not applicable.
	dataType	The data type of the parameter.	OBJECT, INTEGER, STRING, BOOLEAN, DATETIME, DOUBLE, LONG, FLOAT	No	String
	format	Used to format parameters with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

Parameters Example

```
<Parameters>
    <Parameter name="UnivHdr" value="SAMPLE"/>
    <Parameter name="Feed" value="ProdServer1"/>
    <Parameter name="LogDate" dataType="DATETIME" value="%LogDate%" format="yyyyMMdd"/>
    <Parameter name="FiscalYear" dataType="INTEGER" value="2008"/>
</Parameters>
```

4. The next section of the `CollectorInput` stage is the `InputFields` element. The `InputFields` element defines the input. Each field is defined as a sub-element of the `InputFields` element. This section is required.

   The following tables describe the applicable elements and attributes for the `InputFields` element by collector type: DELIMITED, FIXEDFIELD, REGEX, or DATABASE..

Table 61. Elements and attributes specific to `InputFields` for the `DELIMITED` collector

Element	Attribute	Description	Values	Required	Default
InputField		Used to define an input field. (Required: 1 or more elements)			

Table 61. Elements and attributes specific to `InputFields` for the `DELIMITED` collector (continued)

Element	Attribute	Description	Values	Required	Default
	name	A unique name to identify the input field.	Any value.	Yes	Not applicable.
	position	The column position in the input.	A positive integer.	Yes	Not applicable.
	dataType	The data type of the input field.	OBJECT, INTEGER, STRING, BOOLEAN, DATETIME, DOUBLE, LONG, FLOAT	Yes	Not applicable.
	format	Used to format input fields with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

InputFields for DELIMITED Example

```
<InputFields>
  <InputField name="Column1" position="1" dataType="DATETIME" format="MMddyyyy"/>
  <InputField name="Column2" position="2" dataType="DATETIME" format="HH:mm"/>
  <InputField name="Column3" position="3" dataType="STRING"/>
  <InputField name="Column4" position="4" dataType="INTEGER"/>
</InputFields>
```

Table 62. Elements and attributes specific to `InputFields` for the `FIXEDFIELDS` collector

Element	Attribute	Description	Values	Required	Default
InputField		Used to define an input field. (Required: 1 or more elements)			
	name	A unique name to identify the input field.	Any value.	Yes	Not applicable.
	startingColumn	The starting column position of the input field in the input file.	A positive integer.	Yes	Not applicable.
	length	The length of the input field in the input file.	A positive integer.	Yes	Not applicable.
	dataType	The data type of the input field.	OBJECT, INTEGER, STRING, BOOLEAN, DATETIME, DOUBLE, LONG, FLOAT	Yes	Not applicable.
	format	Used to format input fields with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

InputFields for FIXEDFIELDS Example

```
<InputFields>
  <InputField name="Column1" startingColumn="1" length="5" dataType="DATETIME" format="MMddyyyy"/>
  <InputField name="Column2" startingColumn="6" length="10" dataType="DATETIME" format="HH:mm"/>
  <InputField name="Column3" startingColumn="17" length="23" dataType="STRING"/>
  <InputField name="Column4" startingColumn="41" length="3" dataType="INTEGER"/>
</InputFields>
```

Table 63. Elements and attributes specific to `InputFields` for the `REGEX` collector

Element	Attribute	Description	Values	Required	Default
InputField		Used to define an input field. (Required: 1 or more elements)			
	name	A unique name to identify the input field.	Any value.	Yes	Not applicable.
	position	The column position in the input.	A positive integer.	Yes	Not applicable.
	dataType	The data type of the input field.	OBJECT, INTEGER, STRING, BOOLEAN, DATETIME, DOUBLE, LONG, FLOAT	Yes	Not applicable.
	format	Used to format input fields with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

InputFields for REGEX Example

```
<InputFields>
  <InputField name="Field1" position="1" dataType="DATETIME" format="MM/dd/yyyy"/>
  <InputField name="Field2" position="2" dataType="INTEGER"/>
  <InputField name="Field3" position="3" dataType="STRING"/>
  <InputField name="Field4" position="4" dataType="STRING"/>
</InputFields>
```

Table 64. Elements and attributes specific to `InputFields` for the `DATABASE` collector

Element	Attribute	Description	Values	Required	Default
InputField		Used to define an input field. (Required: 1 or more elements)			
	name	A unique name to identify the input field.	Any value.	Yes	Not applicable.
	position	The column position in the input.	A positive integer.	Yes	Not applicable.
	columnName	The name of the column in the database.	A valid name.	Yes(You must define a position or a columName attribute.)	Not applicable.
	dataType	The data type of the input field.	OBJECT, INTEGER, STRING, BOOLEAN, DATETIME, DOUBLE, LONG, FLOAT	Yes	Not applicable.
	format	Used to format input fields with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

InputFields for DATABASE Example

```
<InputFields>
  <InputField name="Column1" columnName="name" dataType="DATETIME" format="MMddyyyy"/>
  <InputField name="Column2" columnName="capacity" dataType="DATETIME" format="HH:mm"/>
  <InputField name="Column3" columnName="disks" dataType="STRING"/>
  <InputField name="Column4" columnName="serialNo" dataType="INTEGER"/>
</InputFields>
```

5. The next section of the `CollectorInput` stage is the `QueryParameters` element. The `QueryParameters` element defines a database query and contains sub-elements which define the database connection, database SQL statement, parameters, and result set. This section is optional.

   The following table describes the applicable elements and attributes for the `QueryParameters` element.

Table 65. Elements and attributes specific to `QueryParameters`

Element	Attribute	Description	Values	Required	Default
Connection		Used to define the Data Source Name. (Required: 1 element only)			
	dataSourceName	A Data Source Name defined on the Data Source List Maintenance page in Administration Console.	A valid data source name.	Yes	Not applicable.
Statement		Used to define the SELECT statement or a stored procedure. (Required: 1 element only)			
	type	The statement type (SQL statement or a stored procedure).	PROCEDURE, SQL	No	SQL
	text	A SQL statement or a stored procedure name. Use the ? character to represent placeholders for the parameters.	A valid SQL statement or stored procedure name.	Yes	Not applicable.
Parameter		Used to define the parameters for the SQL statement or stored procedure. It is not required if the SQL statement or stored procedure is not accepting any parameters. (Optional: 0 or more elements)			
	src	The parameter can be a static value or come from a parameter in the parameters collection	STATIC, PARAMETER	Yes(You must define either the value attribute, or the src and srcName attributes, with src set to STATIC.)	STATIC

Table 65. Elements and attributes specific to `QueryParameters` (continued)

Element	Attribute	Description	Values	Required	Default
	sqlType	The SQL type of the statement parameter.	STRING, VARCHAR, CHAR, INTEGER, LONG, DOUBLE, FLOAT, DATE, TIME, TIMESTAMP, BOOLEAN	Yes	Not applicable.
	position	The position of the parameter in the SQL statement or stored procedure signature.	A positive integer.	Yes	Not applicable.
	value	A value for the STATIC parameter.	Any value.	Yes, if src is STATIC.	Not applicable.
	srcName	The name of the source parameter or query parameter.	A name which is defined in the Parameters or QueryParameters section.	Yes, if src is not STATIC.	Not applicable.
	format	Used to format parameters with sqlType of DATE, TIME, and TIMESTAMP.	A valid Java date format string.	No	Not applicable.
ResultField		Used to define the result set returned from the database.			
	name	A name to identify the result field.	Any unique name.	Yes	Not applicable.
	position	The position of the column in the result set.	A positive integer.	Yes(You must define either the position attribute or the columnName attribute.)	Not applicable.
	columnName	The column name in the result set.	A valid column name	Yes	Not applicable.
	dataType	The data type of the result field	DATETIME, STRING, INTEGER, LONG, BOOLEAN, FLOAT, OBJECT	Yes	Not applicable.
	format	Used to format input fields with dataType of DATETIME.	A valid Java date format string.	No	Not applicable.

QueryParameters Example

```
<QueryParameters>
  <Connection dataSourceName="DBCollector"/>
  <Statement text=" SELECT * FROM CIMSTransaction WHERE (ToDate &gt;= ? AND ToDate &lt;= ?)"/>
  <Parameter src="STATIC" position="1" sqlType="DATE" value="01/01/2007" format="M/d/yyyy"/>
  <Parameter src="STATIC" position="2" sqlType="DATE" value="01/31/2007" format="M/d/yyyy"/>
  <ResultField name="Account_code" position="4" dataType="STRING"/>
  <ResultField name="ServerName" position="5" dataType="STRING"/>
  <ResultField name="Instance" position="6" dataType="STRING"/>
</QueryParameters>
```

6. The next section of the `CollectorInput` stage is the `OutputFields` element. The `OutputFields` element defines the output. Each field is defined as a sub-element of the `OutputFields` element. This section is required.

   The following table describes the applicable elements and attributes for the `OutputFields` element.

Table 66. Elements and attributes specific to `OutputFields`

Element	Attribute	Description	Values	Required	Default
OutputField		Used to define an output field			
	name	A unique name to identify the column name in the output file (refer to the information that follows this table).	Any value.	Yes	Not applicable.
	src	The section whose input will map to the output field.	INPUT, PARAMETER, QUERYPARAMETER, KEYWORD	Yes	Not applicable.
	srcName	The name of the field in the section referred to by src.	A valid name.	Yes, if src is not KEYWORD.	Not applicable.
	dateKeyword	Defines usage dates by using system date keywords.	SYSDATE, RUNDATE, PREDAY, PRECURDAY, PREWEEK, PREMON, CURWEEK, CURMON	Yes, if src is not KEYWORD.	Not applicable.
	timeKeyword	Defines usage times by using system time keywords.	SYSTIME, ENTIREDAY	Yes, if dateKeyword is SYSDATE.	Not applicable.
	resource	Defines whether this output field is a resource, or alternatively, an identifier.	yes, no	No	The output field is an identifier.

The following names have special meaning:

- headerrectype – Record Source
- headeraccountcode – Account Code
- headershiftcode – Shift Code
- headerstartdate – Usage Start Date
- headerstarttime – Usage Start Time
- headerenddate – Usage End Date
- headerendtime – Usage End Time

Giving a dateKeyword is sufficient for specifying HeaderStartDate, HeaderStartTime, HeaderEndDate, and HeaderEndTime (timeKeyword will be required if dateKeyword is SYSDATE). The name of the output field for which the dateKeyword is specified does not matter.

OutputFields Example

```
<OutputFields>
  <OutputField name="headerrectype" src="parameter" srcName="Resourceheader"/>
  <OutputField name="headerUsageDates" src="keyword" dateKeyword="SYSDATE" timeKeyword="ENTIREDAY"/>
  <OutputField name="Feed" src="parameter" srcName="Feed"/>
  <OutputField name="User" src="input" srcName="1"/>
  <OutputField name="Server" src="input" srcName="2"/>
  <OutputField name="RR01" src="input" srcName="3" resource="no"/>
</OutputFields>
```

If you want to have some header usage dates come from a date keyword while others are mapped to an input source, you can define the output field with src=”keyword” first and then define output fields for the usage dates coming from an input source. The later fields will overwrite the effects of the previous field. Example:

```
<OutputFields>
  <OutputField name="headerrectype" src="PARAMETER" srcName="Resourceheader"/>
  <OutputField name="headerUsageDates" src="KEYWORD" dateKeyword="SYSDATE" timeKeyword="ENTIREDAY"/>
  <OutputField name="headerstarttime" src="INPUT" srcName="field5Time"/>
```

```
  <OutputField name="headerenddate" src="INPUT" srcName="field4Date"/>
  <OutputField name="Feed" src="PARAMETER" srcName="Feed" />
  <OutputField name="User" src="INPUT" srcName="1" />
  <OutputField name="Server" src="INPUT" srcName="2" />
  <OutputField name="RR01" src="INPUT" srcName="3" resource="no"/>
</OutputFields>
```

7. The next section of the `CollectorInput` stage is the `Files` element. The `Files` element defines the files which will be used in the collection process. This section is required unless the collector being used is `DATABASE`. However, if doing exception processing, `DATABASE` also requires the `Files` section.

   The following table describes the applicable elements and attributes for the `Files` element.

Table 67. Elements and attributes specific to `Files`

Element	Attribute	Description	Values	Required	Default
File		Used to define a file name, type. (Required: 1 or more elements)			
	name	The name of the file .	Full path to the file.	Yes	Not applicable.
	type	Defines whether the file will be used for collection or for writing out exceptions.	input, exception	No	input

Files Example

```
<Files>
  <File name="<SCCM_install_dir>\CollectorLogs\SodaLog.txt" type="input"/>
  <File name="<SCCM_install_dir>\CollectorLogs\exception.txt" type="exception"/>
</Files>
```

Creating new collectors using Java

Use this topic to create collectors using Java. The use of Java to create collectors is for expert users only.

Procedure

1. Create a new Java class which extends one of the generic collector classes (DATABASE, DELIMITED, FIXEDFIELD) or one of the product collector classes. For example: `public class TPC extends DELIMITED`

   The class must be in package `com.ibm.tivoli.tuam.integrator.collector.`
2. Create a default constructor that throws the `DataInvalidException` and `DataWarningException` For example, `public TPC() throws DataInvalidException, DataWarningException`
3. The following four methods can be overridden with collector specific implementation requirements.
   - public void processInput() throws DataInvalidException, DataWarningException
   - public void validateInput() throws DataInvalidException, DataWarningException
   - public boolean readRecord() throws CollectorException
   - public void processRecord(Fields fields) throws DataInvalidException

   Note: The `public boolean readRecord() throws CollectorException` method can only be overridden in the database collector.

4. The `processInput` method is used to define the details of the input, input fields, parameters and query parameters. The generic collector you extend will determine which collection you will use to define specific related input. The following is the relationship between generic collector and collection: `extends DATABASE: inputParameters.databaseSettings ; DELIMITED: inputParameters.delimitedSettings ; FIXEDFIELD: inputParameters.fixedFieldSettings.`

   The `inputParameters.inputFields` collection is used to define the input fields of the input for the collector.

   The `inputParameters.parameters` collection is used to define the parameters for the collector, for example, Log Date, Feed, Resource Header).

   The `inputParameters.queryParameter` collection is used to define a database query which will return back a single row of data. The column data can be used as parameters.
5. The `validateInput` method is used to insure the `xxxSettings`, `inputFields`, and `parameters` collections are valid per the requirements of the collector. It also allows you to modify the various collections with data that might not have been available when the framework called the `processInput` method. This method is called by the framework prior to the method which processes the input. An example implementation might be the following: Your collector is going to query a database based off a data range in calendar period in the SmartCloud Cost Management calendar. Your job file has defined a parameter which is a calendar period. You extract the calendar period value from the parameters collection and query the calendar table to return the date range. You pass the date range as parameters into your SQL statement.
6. The `readRecord` method can be used to add EOF Processing. The method will first need to call its super method to determine the EOF.
7. The `processRecord` method is where all the record-level business logic is implemented for the collector. This method exposes the current record. You have the ability to edit the record. If you want to process the record, make a call to its super method, otherwise the record is not processed.
8. The framework calls the methods and processes the job file in the following order:

   The following are called only once.

   - processInput method which processes the job file and overrides any parameters found in code with the parameters in the job file.
   - validInput method

   The following are called for each record in the input, unless any of the previously mentioned methods return an exception.

   - readRecord method
   - processRecord method

IBM SmartCloud Cost Management Core data collectors

This section describes the Core data collectors provided in IBM SmartCloud Cost Management.

AIX Advanced Accounting data collector

The AIX Advanced Accounting data collector accumulates input from supported record types and generates log files.

To collect data for AIX Advanced Accounting, you must set up Advanced Accounting data files and data collection on an AIX system as described in "Setting up AIX Advanced Accounting data collection" on page 397. On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces log files. There is a separate log file for each of the supported record types in the data files. The record types are:

- 1 or 2 Process records
- 4 System processor and memory interval record
- 6 File system activity interval record
- 7 Network interface I/O interval record
- 8 Disk I/O Interval record
- 10 Virtual I/O Server interval record
- 11 Virtual I/O Client interval record
- 16 Aggregated ARM transaction record
- 36 WPAR System interval record
- 38 WPAR File System activity interval record
- 39 WPAR Disk I/O interval record

The Advanced Accounting log files that are created on an AIX system can be sent to the SmartCloud Cost Management application server on Windows for processing. The log files are placed in the appropriate `<SCCM_install_dir>/logs/collector/AACCT_n/<feed>` directory, where `n` specifies the Advanced Account record type and `feed` specifies the log file source. For example, log files containing Advanced Accounting data for record type 1 from the server `zeus` are in the `...logs/collectors/AACCT_1/zeus` subdirectory.

To set up data collection for Advanced Accounting on an AIX system and to send the files to a Windows system, use the scripts and files provided for the AIX Advanced Accounting Data Collector for Linux and UNIX.

Process data collected (Advanced Accounting record type 1)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for process records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- PROJID (project ID)

- UserName
- Group
- ProcessName

Resources

- AAID0101 (AIX Process Interval Count)
- AAID0102 (AIX Process Elapsed Time [seconds])
- AAID0103 (AIX Process Elapsed Thread Time [seconds])
- AAID0104 (AIX Process CPU Time [seconds])
- AAID0105 (AIX Elapsed Page Seconds Disk Pages)
- AAID0106 (AIX Elapsed Page Seconds Real Pages)
- AAID0107 (AIX Elapsed Page Seconds Virtual Memory)
- AAID0108 (AIX Process Local File I/O [MB])
- AAID0109 (AIX Process Other File I/O [MB])
- AAID0110 (AIX Process Local Sockets I/O [MB])
- AAID0111 (AIX Process Remote Sockets I/O [MB])

System data collected (Advanced Accounting record type 4)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for system processor and memory interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour

Resources

- AAID0401 (AIX System Number of CPUs [interval aggregate])
- AAID0402 (AIX System Entitled Capacity [interval aggregate])
- AAID0403 (AIX System Pad Length [interval aggregate])
- AAID0404 (AIX System Idle Time [seconds])
- AAID0405 (AIX System User Process Time [seconds])
- AAID0406 (AIX System Interupt Time [seconds])
- AAID0407 (AIX System Memory Size MB [interval aggregate])
- AAID0408 (AIX System Large Page Pool [MB])
- AAID0409 (AIX System Large Page Pool [MB in-use])
- AAID0410 (AIX System Pages In)
- AAID0411 (AIX System Pages Out)
- AAID0412 (AIX System Number Start I/O)
- AAID0413 (AIX System Number Page Steals)
- AAID0414 (AIX System I/O Wait seconds)

- AAID0415 (AIX System Kernal (System) CPU seconds)
- AAID0416 (AIX System Interval Elapsed seconds)

Note: In order to use the AIX AA collector to collect data for resources AAID0414, AAID0415, and AAID0416 on AIX version 6.1 or later, the readaacct utility is required. Details of how the readaacct utility can be configured for use can be found on the IBM SmartCloud Cost Management Technical Support web site.

File system data collected (Advanced Accounting record type 6)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for file system activity interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- FS_TYPE (file system type)
- Device
- MOUNT_PT (mount point)

Resources

- AAID0601 (AIX FS Bytes Transferred [MB])
- AAID0602 (AIX FS Read/Write Requests)
- AAID0603 (AIX FS Number Opens)
- AAID0604 (AIX FS Number Creates)
- AAID0605 (AIX FS Number Locks)

Network data collected (Advanced Accounting record type 7)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for network interface I/O interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour

- Interface

Resources

- AAID0701 (AIX Network Number I/O)
- AAID0702 (Network Bytes Transferred [MB])

Disk data collected (Advanced Accounting record type 8)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for disk I/O interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- DISKNAME

Resources

- AAID0801 (AIX Disk Transfers)
- AAID0802 (AIX Disk Block Reads)
- AAID0803 (AIX Disk Block Writes)
- AAID0804 (AIX Disk Transfer Block Size [interval])

Virtual I/O server data collected (Advanced Accounting record type 10)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for server VIO interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- SERPARNO (server partition number)
- SERUNID (server unit number)
- DLUNID (device logical unit ID)

Resources

- AAID1001 (Virtual I/O Server Bytes In)
- AAID1002 (Virtual I/O Server Bytes Out)

Virtual I/O client data collected (Advanced Accounting record type 11)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for client VIO interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- SERPARNO (server partition number)
- SERUNID (server unit number)
- DLUNID (device logical unit ID)

Resources

- AAID1101 (Virtual I/O Client Bytes In)
- AAID1102 (Virtual I/O Client Bytes Out)

ARM transaction data collected (Advanced Accounting record type 16)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for aggregated ARM transaction records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- PROJID
- APP_CLASS
- APP_NAME
- USERNAME
- GROUP
- TRANSACTION

Resources

- AAID1601 (AIX Application Count)
- AAID1602 (AIX Response Time [seconds])
- AAID1603 (AIX Queued Time [seconds])
- AAID1604 (AIX Application CPU Time [seconds])

WPAR system data collected (Advanced Accounting record type 36)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for WPAR system interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources *are not* pre-loaded in the Rate table. You can use Administration Console to load the rate codes into the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- wparname

Resources

- AAID3601 (Number of CPUs [Global interval value])
- AAID3602 (Entitled capacity [Global interval value])
- AAID3603 (System pad length [Global interval value])
- AAID3604 (System idle time [Unused])
- AAID3605 (User process time in seconds)
- AAID3606 (Interrupt time [Unused])
- AAID3607 (Memory size in MB [Global interval value])
- AAID3608 (Large page pool in MB [Unused])
- AAID3609 (Large page pool in-use in MB [Unused])
- AAID3610 (Pages in)
- AAID3611 (Pages out)
- AAID3612 (Number start I/O)
- AAID3613 (Number page steals)
- AAID3614 (I/O wait time [Unused])
- AAID3615 (Kernel process time in 1/1000 seconds)
- AAID3616 (Interval elapsed time in 1/1000 seconds)
-

WPAR file system data collected (Advanced Accounting record type 38)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for WPAR file system activity interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources *are not* pre-loaded in the Rate table. You can use Administration Console to load the rate codes into the Rate table.

Identifiers

- SYSTEM_ID
- Sysid

- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- FS_TYPE (file system type)
- Device
- MOUNT_PT (mount point)
- wparname

Resources

- AAID3801 (Bytes transferred in MB)
- AAID3802 (Read/Write requests)
- AAID3803 (Number opens)
- AAID3804 (Number creates)
- AAID3805 (Number locks)
-

WPAR disk data collected (Advanced Accounting record type 39)

On an AIX system, the AIX Advanced Accounting Data Collector gathers data from Advanced Accounting data files and produces separate log files for WPAR disk I/O interval records. The data is defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources *are not* pre-loaded in the Rate table. You can use Administration Console to load the rate codes into the Rate table.

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- DISKNAME
- wparname

Resources

- AAID3901 (Network number I/O)
- AAID3902 (Network bytes transferred in MB)
-

Setting up AIX Advanced Accounting data collection

This topic provides information about setting up AIX Advanced Accounting data collection.

The Advanced Accounting data collector uses the Integrator program to convert data in the Advanced Accounting log file into a CSR or CSR+ file. The Integrator program is run from a job file and uses the common XML architecture used for all data collection in addition to elements that are specific to Integrator.

SmartCloud Cost Management includes a sample job file, `SampleAIXAA.xml`, that you can modify and use to process any Advanced Accounting usage log. After you have modified and optionally renamed the `SampleAIXAA.xml` file, move the file to the `<SCCM_install_dir>/jobfiles` directory.

The collector type is defined within the sample job file as an AIX Advanced Accounting collector using the following piece of code:

```
<Input name="AIXAAInput" active="true">
 <Files>
  <File name="... " />
  ...
 </Files>
</Input>
```

HPVMSar data collector

HPVMSar is a command available on the HP Integrity VM platforms, which provide the user with system-level CPU utilization and capacity information.

HPVMSar reports information in time intervals as requested by the user. For example, the user may request hpvmsar to report 10 5-minute intervals of utilization and capacity information.

```
# hpvmsar –s 300 –n 10
```

The SmartCloud Cost Management hpvmsar collector provides a script which is scheduled in root's crontab to generate hpvmsar reports and transfer the reports to the SmartCloud Cost Management Server on a daily basis for processing. The hpvmsar reports are processed using a SmartCloud Cost Management jobrunner job file. SmartCloud Cost Management Server provides the following files to implement the SmartCloud Cost Management hpvmsar Collector

```
.../collectors/hpvmsar/hpvmsar_collect.template
.../collectors/hpvmsar/tuam_unpack_hpvmsar
.../collectors/hpvmsar/hpvmsarDeploymentManifest_hp.xml
.../samples/jobfiles/samples/jobfiles/SampleDeployHPVMSarCollector.xml
.../samples/jobfiles/samples/jobfiles/SampleHPVMSar.xml
```

Identifiers and resources collected by hpvmsar collector

Identifiers and resources collected by the hpvmsar collector are converted to CSR record format by an Integrator stage in the SampleHPVMSar.xml jobfile.

The following identifiers and resources are included in the CSR file.

Identifiers:

Table 68. HPVMSar Identifiers

Identifier	Description
HOST_NAME	Name of the host server

Table 68. HPVMSar Identifiers (continued)

Identifier	Description
HOST_IP_ADDR	IP address of host server
VM_NAME	Name of the virtual machine
VM_IP_ADDR	IP address of the virtual machine
INT_START	Interval start time YYYYMMDDHHMMSS
INT_START	Interval end time YYYYMMDDHHMMSS

Resources:

Table 69. HPVMSar Resources

Resource	Description
HSNUMCPU	Number of virtual CPUs of the virtual machine
HSPHYMEM	Physical memory size (MB)
HSMEMSIZ	Virtual machine memory size (MB)
HSCPUUPT	CPU usage percentage of the virtual machine

It is recommended that this data is written to the resource utilization table. For more information on how to do this, see the DBLoad section.

Deploying the SmartCloud Cost Management hpvmsar collector

The SmartCloud Cost Management hpvmsar Collector can be deployed in one of two ways. First, using the RXA capability built into SmartCloud Cost Management Server. This method allows the user to install the hpvmsar collector to the target server from the SmartCloud Cost Management Server. A second method would be to manually copy the hpvmsar installation files to the target server and run the tuam_unpack_hpvmsar installation script.

Regardless of the method used to install, the collector requires that the SmartCloud Cost Management Server is running an FTP service to receive the nightly hpvmsar usage files from the target servers. Where SmartCloud Cost Management server is running on Windows, you will need to be sure of the following:

- The IIS FTP service is installed.
- A virtual directory has been defined to the SmartCloud Cost Management Collector Logs folder.
- The FTP service can be configured to allow either anonymous or user/password access, but the accessing account must have read/write access to the SmartCloud Cost Management Collector Logs folder.

Installing the SmartCloud Cost Management hpvmsar Collector from the SmartCloud Cost Management Server:

Installing the hpvmsar Collector from the SmartCloud Cost Management Server is accomplished using the `SampleDeployHPVMSarCollector.xml` jobfile.

Before you begin

Several parameter settings need to be edited prior to running the jobfile.

- **Host**: The hostname or IP address of the target server.
- **UserId**: Must be set to "root".

- **Password**: Password for root on the target server.
- **Manifest**: Manifest file for the OS type of the target server.
- **RPDParameters**: See note in the example.

Example

Example from `SampleDeployHPVMSarCollector.xml` jobfile:

Note: Example lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
                <!-- SUPPLY hostname OF TARGET PLATFORM/-->
                <Parameter Host = "TARGET_PLATFORM"/>

                <!-- userid must be set to root/-->
                <Parameter UserId = "root"/>

                <!-- SUPPLY root PASSWORD ON TARGET PLATFORM/-->
                <Parameter Password = "XXXXXX"/>

                <!--Parameter KeyFilename = "yourkeyfilename"/-->

                <!-- DEFINE Manifest TO MANIFEST XML FOR TARGET PLATFORM/-->
                <!--Parameter Manifest = "hpvmsarDeploymentManifest_hp.xml"/-->
                <Parameter Manifest = "hpvmsarDeploymentManifest_hp.xml"/>

                <!--Parameter Protocol = "win | ssh"/-->

                <!-- DEFINE INSTALLATION PARAMETERS,
                       path:   must be defined to the directory path where hpvmsar Collector
                               will be installed on target platform.
                               Parameter Required.
                       server: name or IP address of the SmartCloud Cost Management Server
                               Parameter Required.
                       log_folder: CollectorLog folder on
SmartCloud Cost Management Server.
If SmartCloud Cost Management Server
is UNIX/Linux platform, log_folder should be defined to the full path to the
                               Collector Logs folder, example: /opt/ibm/tuam/logs/collectors .
                               If SmartCloud Cost Management Server is Windows platform, log_folder should be set to
                               virtual directory that references the Collector Logs folder.
                               Parameter Required.
                       ftp_user: Account used to access the SmartCloud Cost Management Server for nightly transfer of the
                               hpvmsar log to the SmartCloud Cost Management Server.  If the log file is to remain on the
                               client, set ftp_user=HOLD. If anonymous ftp access has been configured
                               on the SmartCloud Cost Management Server, enter ftp_user=anonymous. This account must have
                               read/write access to the Collector Logs folder.
                               Parameter Required.
                       ftp_key: password used by ftp_user. If ftp_user=anonymous or ftp_user=HOLD,
                               enter any value for this parameter, but enter something,
                               example: ftp_key=XXXX.
                               Parameter Required.
                       add_ip: Indicate if the name of the folder receiving the hpvmsar files on
                               the SmartCloud Cost Management Server should include the IP address of the client server.
                               example: If log_folder=/opt/ibm/tuam/logs/collectors and the
                               client server  is named "client1", the nightly hpvmsar file will be
                               placed in
                                   /opt/ibm/tuam/logs/collectors/hpvmsar/client1
                               If add_ip=Y, the file will be delivered to
                                  /opt/ibm/tuam/logs/collectors/hpvmsar/client1_<ipaddress>.
                               Default value "N".
                       interval: Number of seconds in hpvmsar intervals.
                               Default value 300.
                -->
                <Parameter RPDParameters = "path=/data/tuam/collectors/hpvmsar;server=9.42.17.133;\
                log_folder=/data/tuam/logs/collectors;ftp_user=ituam;ftp_key=ituam;add_ip=Y;interval=300;"/>
```

Manually installing the hpvmsar collector:

This topic describes how you can manually install the hpvmsar collector.

About this task

The SmartCloud Cost Management hpvmsar Collector can be installed manually by copying the collectors installation files to the target server, placing them in the directory where you want the collector to reside and then executing the collector unpack script.

For example, suppose your SmartCloud Cost Management Server is installed in <SCCM_install_dir>, and you want the SmartCloud Cost Management hpvmsar collector to reside in /opt/ibm/tuam/collectors/hpvmsar on the target server, to complete the installation you would do the following:

1. Copy <SCCM_install_dir>\collectors\hpvmsar_collect.template to /opt/ibm/tuam/collectors/hpvmsar on the target server.
2. Copy <SCCM_install_dir>\collectors\tuam_unpack_hpvmsar to /opt/ibm/tuam/collectors/hpvmsar on the target server.
3. On the target server, as root, enter the following. (Note: the values supplied are described in the installing topic).

   Note: Code lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
# cd /opt/ibm/tuam/collectors/hpvmsar
# chmod 770 tuam_unpack_hpvmsar
# ./tuam_unpack_hpvmsar  path=/opt/ibm/tuam/collectors/hpvmsar  server=tuamserver \
log_folder=collector_log  ftp_user=ituam  ftp_key=ituam  add_ip=N  interval=300
```

Following the installation of the hpvmsar collector:

This topic discusses the actions that must be taken after you have installed the hpvmsar collector.

The installation places an entry in the root crontab to run the hpvmsar_collect.sh script once each hour. The script will output an hourly hpvmsar report to `/<SCCM_install_dir>/data/hpvmsar_YYYYMMDD_HH.txt`.

When called in the 23rd hour (11 PM), the script will generate the hourly report and then concatenate the hourly hpvmsar reports into a single file `hpvmsar_YYYYMMDD.txt`. This file is then transferred to the SmartCloud Cost Management Server and placed in `<collector_logs>/hpvmsar/<target_name>`. A log of the FTP transfer is written to `/<SCCM_install_dir>/data/YYYYMMDD_ftp.log`

The daily hpvmsar usage files and ftp logs are retained on the target platform for 10 days.

Script - hpvmsar_collect.sh

Usage:

```
hpvmsar_collect.sh  collect  |  send [YYYYMMDD ]
```

The `hpvmsar_collect.sh` script is called with either the collect or send argument.

collect : This argument should only be used in the crontab entry that is placed in root's cron file when the hpvmsar collector is installed.

send : This argument can be used to send the nightly hpvmsar report file to the SmartCloud Cost Management Server. A log of the FTP transfer is written to `/<SCCM_install_dir>/data/YYYYMMDD_ftp.log`.

Uninstalling the hpvmsar collector:

This topic describes how to uninstall the hpvmsar collector.

About this task

The SmartCloud Cost Management hpvmsar Collector can be uninstalled simply by removing the `hpvmsar_collect.sh` crontab entry from root's crontab. Once that's done, all hpvmsar collector files and scripts can be removed.

OpenStack data collector

The OpenStack data collector collects virtual machine (VM) instance utilization data from OpenStack Nova Compute, Cinder Volume, and VM context data from OpenStack Keystone. This provides the complete picture of the utilization of VMs and who the VMs are allocated to for a time period.

The data collector consists of two separate collectors:

- Metering Control Service (MCS)
- Keystone collector

The MCS collects system usage data from OpenStack components Nova Compute, and Cinder Volume. The data is emitted in the form of events from OpenStack's Notification System. The MCS handles immediate and periodic events (which are published on an hourly basis) to an Advanced Message Queuing Protocol (AMQP) broker message queue. The MCS processes the events and generates SmartCloud Cost Management CSR records on a daily basis. The processed daily data is then loaded into the SmartCloud Cost Management database, allowing reports to run based on historical data.

The Keystone collector collects user, project, and domain data from the Keystone Identity Service API. The Keystone Identity Service API maintains user, project, domain, and role data and it allows clients to obtain tokens that are used to access OpenStack cloud services. The Keystone collector has two main functions:

1. The collector collects user, project, and domain data for all OpenStack clouds that are managed by the Keystone instance. The data that is produced by the collector is loaded into the SmartCloud Cost Management database as conversion mappings with the utilization data. These conversion mappings are then used to map meaningful context identifiers into the utilization data.
2. The collector can also be configured to produce account code security related identifiers based on user roles in Keystone. These identifiers are then used by the `CreateAccountRelationship` and `CreateUserRelationship` stages so users, user groups, and clients can be automatically created in the SmartCloud Cost Management database.

Configuring the OpenStack data collector

Use the topics in this section to configure the OpenStack data collector.

Important: The OpenStack data collector is automatically configured by running the ico_configure.sh script after SmartCloud Cost Management is installed. For more information about this script, see the related topic.

Refer to the sub topics in this section if you want to manually configure OpenStack data collector for metering.

Manually configuring the OpenStack data collector:

Use the topics in this section to manually configure the OpenStack data collector.

Configuring IBM Cloud Orchestrator for metering:

Each IBM Cloud Orchestrator region must be configured to generate OpenStack notifications which are the raw input data for SmartCloud Cost Management metering. For more information, see https://wiki.openstack.org/wiki/SystemUsageData.

This manual configuration is controlled by the `<SCCM_install_dir>/bin/postconfig/enable_openstack_notifications.sh` script. The script performs the following steps:

1. It identifies each region and adds entries to Cinder volume configurations to enable notification generation on that region.
2. It enables network connectivity between the Metering Control Service and the OpenStack message brokers. Each OpenStack region has its own message broker instance.

The script is run with the following arguments which enables metering for all IBM Cloud Orchestrator regions and any compute nodes for those regions:

```
<SCCM_install_dir>/bin/postconfig/enable_openstack_notifications.sh \
   --host=<ICO-Keystone-host> --password=<broker-password> [--type=<broker-type>]
```

where

<ICO-Keystone-host>
: Is the host name of the IP address of the Keystone server in the IBM Cloud Orchestrator environment.

<broker-password>
: Is the password for the message broker. By default, it is the Keystone `admin` user password.

<broker-type>
: Is the OpenStack message broker type. It can be `qpid` or `rabbitmq`. The default value is `rabbitmq`.

The `enable_openstack_notifications.sh` script can be rerun without any adverse effect. It might be necessary to update the connectivity parameters if not using a default IBM Cloud Manager with OpenStack installation.

Note: The default message broker for IBM Cloud Manager with OpenStack is RabbitMQ, while other OpenStack distributions may use Apache Qpid. Also if the message broker credentials are changed, ensure that connectivity to the region message brokers is still working.

Procedure

1. In the Administration Console, click **System Configuration** > **Data Sources** and select **Message broker** as the **Data Source Type**.
2. Click on each entry in the **Data Source Name** dropdown and ensure that a successful connection can be made.
3. If errors are shown, update the **Data Source Properties** and click **Save**.

The following topics cover enablement for each OpenStack service.

Configuring OpenStack Nova Compute for metering:

Each compute node must be configured to generate system usage data. For more information, see https://wiki.openstack.org/wiki/SystemUsageData.

For metering data to be collected, the following lines need to be present in `/etc/nova/nova.conf`. For the standard IBM Cloud Manager with OpenStack installation, these are automatically configured.

```
instance_usage_audit = True
notification_driver = messaging
notify_on_state_change = vm_and_task_state
instance_usage_audit_period = hour
```

Configuring OpenStack Cinder volume for metering:

Each node running a Cinder service must be configured to generate data.

For metering data to be collected, the following lines need to be present in `/etc/cinder/cinder.conf`. For the standard IBM Cloud Manager with OpenStack installation, the `control_exchange` entry is automatically configured. The `enable_openstack_notifications.sh` script updates the `volume_usage_audit_period` and `notification_driver` entries.

```
notification_driver = messaging
control_exchange = cinder
volume_usage_audit_period = hour
```

It also adds the following `cron` entries to enable the generation of periodic events:

```
0 * * * * (PYTHONIOENCODING=utf-8 /usr/bin/cinder-volume-usage-audit 1>>
   /var/log/cinder/cinder_audit_hourly_`date +\%Y\%m\%d`.log 2>&1)

0 2 * * * /usr/sbin/logrotate /root/sccmlogrotate.conf
```

Configuring the Metering Control Service for OpenStack:

The Metering Control Service (MCS) connection to OpenStack must be configured to enable the collection of OpenStack Nova Compute and Cinder Volume notification events and the generation of SmartCloud Cost Management CSR records.

The MCS provides general purpose event-based handling of metering data. The MCS is deployed as a web server, which uses the IBM Websphere Application Server Liberty version 8.5. The MCS server instance is installed by default with SmartCloud Cost Management to the following location `<SCCM_install_dir>/wlp/usr/servers/mcs`.

Configuring the connection to OpenStack:

The Metering Control Service (MCS) uses an Advanced Message Queuing Protocol (AMQP) listener to collect notification events, such as Nova Compute notifications from the OpenStack RabbitMQ or Apache Qpid message broker.

The MCS is configured during the post-installation of SmartCloud Cost Management to collect events from Nova Compute and Cinder Volume from IBM Cloud Orchestrator. The script `<SCCM_install_dir>/bin/postconfig/create_ico_notification_connections.sh` performs the following steps:

Note: Before this script is run, the IBM Cloud Orchestrator OpenStack environment must be configured for metering.

1. Creates data sources that provide connection details to the IBM Cloud Orchestrator region Qpid message brokers.
2. Configures the MCS to listen to notification events for each data source.

This script is run with the following arguments:

```
<SCCM_install_dir>/bin/postconfig/create_ico_notification_connections.sh \
   --host=<ICO-Keystone-host> --password=<broker-password> [--type=<broker-type>]
```

where

<ICO-Keystone-host>
: Is the host name of the IP address of the Keystone server in the IBM Cloud Orchestrator environment.

<broker-password>
: Is the password for the message broker. By default, it is the Keystone `admin` user password.

<broker-type>
: Is the OpenStack message broker type. It can be `qpid` or `rabbitmq`. The default value is `rabbitmq`.

Use the following manual steps to modify or test the data source that the MCS will use.

Manually configuring connections

You can configure the connections to one or more OpenStack instances in different regions.

Note: OpenStack issues event notifications using the Coordinated Universal Time (UTC) timestamps and therefore all files output by the MCS are named according to UTC timestamps.

The SmartCloud Cost Management server must have connectivity to the IBM Cloud Orchestrator region, in order to listen for data notifications being generated.

1. Create a data source that provides connection details to the Qpid message broker. See the related task topic for information about doing this.
2. Configure the MCS to listen to data events from the data source:

   The MCS is configured using the provider instance file in `<SCCM_install_dir>/wlp/usr/servers/mcs/data/providers.json`. This file contains a list of all the registered providers in the MCS. When the MCS starts, each registered provider is enabled. Each registered provider is defined from a provider type and all provider types known to the MCS are defined in a file for each type as follows:

   `<SCCM_install_dir>/wlp/usr/servers/mcs/resources/providers/*.properties.`

   After the MCS is installed, it loads 2 providers of type `NOVA` and `CINDER` broker. The providers are then configured to connect to a default RabbitMQ message broker for each region (port `5671`) and the notification topics of `nova/notifications.info` and `cinder/notifications.info`.

   The provider instance contains the following properties:

   - **`provider_type`**: Specifies the provider type that the instance is based on "**`provider_type`**" from the provider type file.

- **provider_parameters**: The specific parameters for a provider type. Parameters are specified in the provider type file.
- **provider_name**: Unique name for each provider instance.

The following are specific properties for the providers:

- **datasource_name**: This property references a SmartCloud Cost Management data source entry that securely contains the connection details to the OpenStack message broker.
- **message_destination**: This property references a Topic or Queue name, where the Nova Compute notifications are published. The default is `nova/notifications.info`.

If you want to connect to more OpenStack instances, then you must have a line in the providers cache file for each instance.

Similar to when configuring the connection for a single instance, the default entry in the instances file must be modified or copied to a new instance. For example, one instance per line, to meet the configuration that is required. You must create a SmartCloud Cost Management data source that contains the relevant connection details for each OpenStack instance that you want to connect to. See step 1 for information about how to create the data source. When new provider instances are added, the provider thread pool must be updated also. The property that must be updated is called **max_provider_threads** and it can be set in `<SCCM_install_dir>/wlp/usr/servers/mcs/resources/mcs/mcs.properties`. This value must be set to an integer value greater than the number of instances that are running.

Related concepts:

"Automated configuration" on page 7
Most of the post-installation configuration of SmartCloud Cost Management for IBM Cloud Orchestrator is automated. This automation process is controlled by running the `ico_configure.sh` script as the same user that installed SmartCloud Cost Management.

Configuring the generation of CSR records:

The Metering Control Service (MCS) uses a SmartCloud Cost Management handler to generate CSR records from OpenStack compute events.

The SmartCloud Cost Management handler can be configured using the handler instance file in `<SCCM_install_dir>/wlp/usr/servers/mcs/data/handlers.json`. This file is a persisted cache of all registered handlers in the MCS. The MCS enables each registered handler when data is passed to the handlers for processing. Each registered handler is defined from a handler type and all handlers types known to the MCS are defined in a file per type as follows:

`<SCCM_install_dir>/wlp/usr/servers/mcs/resources/handlers/*.properties`

The MCS installation pre-loads a default handler `sccm_local` of type `SCCM` for the OpenStack broker. The OpenStack broker is configured to handle OpenStack events and generate SmartCloud Cost Management CSR records to a daily file on the file system where the MCS is installed.

The handler instance file contains the following properties:

- **handler_name**: Unique name for each handler instance.
- **metering_system_type**: Specifies the handler type of this instance, based on the `metering_system_type` from the handler type file.

- **handler_parameters**: The specific parameters for a handler type. Parameters must be specified in the handler type file.

The following are specific properties for the sccm_local handler:

- **collector_log_files_path**: The path to where CSR daily file is written to. If the MCS is installed with SmartCloud Cost Management server then the path is: <CollectorLogs>/<RECORD_TYPE_LOWERCASE>/<PROVIDER_ID>. If not, then it will use the value set for this parameter in the instance file and if not set then, it defaults to <SCCM_install_dir>/wlp/usr/servers/mcs/data/<RECORD_TYPE_LOWERCASE>/<PROVIDER_ID>.
- **record_types**: MCS record types that are supported by the handler. By default, the value is set to NOVA_COMPUTE.

Note: CSR file names are formatted as <YYYYMMDD>.txt.

Metering Control Service commands:

You can use Metering Control Service (MCS) commands to control the MCS, for example, stopping and starting.

Note: If message brokers (for example, RabbitMQ) are restarted on an IBM Cloud Orchestrator region, then the MCS should be restarted.

The MCS is deployed as an IBM Websphere Application Server Liberty. The server must be running to process events. The MCS is started after the installation of SmartCloud Cost Management but it must be restarted if it is stopped manually. The system where the MCS is installed on is configured to restart the MCS on startup. Here are some commands for controlling the MCS:

- To start or restart the MCS: <SCCM_install_dir>/bin/startServer.sh mcs
- To stop the MCS: <SCCM_install_dir>/bin/stopServer.sh mcs
- To get the MCS status: <SCCM_install_dir>/wlp/bin/server status mcs

For tracing, logs can be checked at the following location: <SCCM_install_dir>/logs/server. The trace level can be configured by using the file <SCCM_install_dir>/config/logging.properties.

Note: Logging is shared with other server instances, for example, SmartCloud Cost Management, with each log instance denoted with a number at the end, such as 0, 1 and so on.

Configuring the Keystone collector:

This section provides details about setting up SmartCloud Cost Management for Keystone data collection.

Creating a Web Service data source

Attention: If you have already run the ico_configure.sh script as described in the related reference topic, then this task is not required. However, you can use the Data Sources page to check that the details for the Collector – Web Service data source are correct.
You must create a data source in the Administration Console that points to the base URL of the Keystone Identity Service API. The data source is referenced in the Keystone collector job runner file. To create the data source in the Administration Console:

1. Click **System Configuration** > **Data Sources** and select **Web service** as the **Data Source Type**.
2. Click **Create Data Source** and complete the following:

   Note: All fields marked with an * are mandatory and must be completed.

   Data Source Name
   Type the name that you want to assign to the data source.

   Note: The following are invalid characters for a data source name: `"/", "\",'"',":","?","<",">",".","|",".".`

   Username
   Type the web service user ID.

   Password
   Type the web service password.

   URL Type the web service URL as follows, using either the http or https protocol as required:
   `http://<Server Name>:port`

   Web Service Type
   Select **REST** as the web service type.

   Keystore File
   The Keystore file contains the vCenter or REST server certificate that is used for authentication during the secure connection between the collector and the vCenter web service. The password is used to access the file. Enter a valid path to the file.

   Keystore Password
   Type the Keystore password.
3. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the vCenter or REST server is verified. You should see a message at the top of the screen indicating that the connection was successful.
   Click **Cancel** if do not want to create the data source.

Extracting identifiers from REST invocations:

This topic describes the usage metrics that are collected by the Keystone collector.

The KEYSTONE template in `StandardTemplates.xml` in `<SCCM_install_dir>/collectors/Keystone` is used by the `OpenstackKeystoneContext.xml` job file. This template defines what metrics are collected from the domain, project, and user Keystone resources, as well as what account code security related metrics are required to generate from Keystone security roles. The Input fields of the KEYSTONE template contain keys such as **`KEYSTONE_USER`** and **`KEYSTONE_DOMAIN`** to indicate which chained API **`resourcePath`** defined in the `OpenstackKeystoneContext.xml` job file the metrics belong to. The `StandardTemplates.xml` for each template also defines what CSR identifier names to map the collected metrics to.

The Keystone metrics are collected using the Keystone Identity Service API and defined in the Keystone `StandardTemplates.xml` as **`InputFields`** using the following notation in the **`expression`** name attribute:

- The notation begins with either /users*/ or /projects*/ depending on the API being collected from. The * at the end of the resource name indicates that the Keystone collector returns all instances of the request resource, either users or projects that are available.
- The next portion of the notation indicates the name of the field to extract from the request resource. For example, /project*/ name indicates that the project name field must be extracted or the /users*/ id indicates the user ID field must be extracted. These **InputField** names are then mapped to **OutputFields**, where the **OutputField srcName** is the same as the **InputField** names. The **OutputField** name defines what the CSR identifier is when the record is written.

OpenStack identifiers and resources

This section describes the identifier and resources defined from the OpenStack components notification events and from OpenStack Keystone.

Compute identifiers and resources:

This topic describes the identifier and resources defined from the OpenStack Nova Compute notification events and from OpenStack Keystone.

Table 70. Standard fields

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
Start Date of Usage	timestamp/ audit_period_beginning	Date in format YYYYMMDD	Nova `compute.instance.exists` event
End Date of Usage	timestamp/ audit_period_ending	Date in format YYYYMMDD	Nova `compute.instance.exists` event
Start Time of Usage	timestamp/ audit_period_beginning	Time in format HH:MM:SS	Nova `compute.instance.exists` event
End Time of Usage	timestamp/ audit_period_ending	Time in format HH:MM:SS	Nova `compute.instance.exists` event

Table 71. Ownership Identifiers

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
TENANT_ID	tenant_id	Tenant UUID	Nova `compute.instance.exists` event
TENANT_NAME	`{"tenants": [ {"name": "customer-x"} ] }`	Tenant name	Keystone **HTTP GET** /tenants
TENANT_DESCRIPTION	`{"tenants": [ {"description":"None"} ] }`	Tenant description	Keystone **HTTP GET** /tenants

Table 71. Ownership Identifiers (continued)

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
TENANT_ENABLED	`{"tenants": [ {"enabled":true} ] }`	If Tenant is enabled or not	Keystone **HTTP GET** /tenants
USER_ID	user_id	User UUID	Nova `compute.instance.exists` event
USER_NAME	`{"users": [ {"name": "customer-x"} ] }`	User name	Keystone **HTTP GET** /users
USER_EMAIL	`{"users": [ {"email":"None"} ] }`	User email address	Keystone **HTTP GET** /users
USER_ENABLED	`{"users": [ {"enabled":true} ] }`	If user is enabled or not	Keystone **HTTP GET** /users

Table 72. Placement identifiers

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
REGION	Derived from PUBLISHER_ID or PROVIDER_ID	Identifier which signifies which OpenStack instance the events are generated from.	SmartCloud Cost Management OpenStack VM instances job file
PROVIDER_ID	provider_id	OpenStack Broker Identifier	Metering Control Service (MCS) provider configuration file.
PUBLISHER_ID	publisher_id	Identifier which signifies which AMQP broker (OpenStack) instance the events are generated from	Nova `compute.instance.exists` event
HOST	host	Host name of the hypervisor	Nova `compute.instance.exists` event
AVAILABILITY_ZONE	availability_zone	Availability zone name	Nova `compute.instance.exists` event

Table 73. History identifiers

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
VM_CREATED_AT	created_at	Timestamp for when the instance record was created in Nova (YYYY-MM-DD hh:mm:ss)	Nova `compute.instance.exists` event
VM_LAUNCHED_AT	launched_at	Timestamp for when the instance was last launched by hypervisor (YYYY-MM-DD hh:mm:ss)	Nova `compute.instance.exists` event
VM_DELETED_AT	deleted_at	Timestamp for when the instance was deleted (YYYY-MM-DD hh:mm:ss)	Nova `compute.instance.exists` event
USEDURN	time between (audit_period_ending, audit_period_beginning)	Periodic interval of the event (Seconds)	Nova `compute.instance.exists` event

Table 74. Virtual Machine identifiers

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
VM_ID	instance_id	Nova instance ID of this instance	Nova `compute.instance.exists` event
VM_NAME	display_name	User selected display name for instance.	Nova `compute.instance.exists` event
VM_STATUS	state	Current state of instance, such as “active” or “deleted”.	Nova `compute.instance.exists` event
VM_STATUS_TEXT	state_description	Additional human readable description of current state of instance	Nova `compute.instance.exists` event
VM_INSTANCE_TYPE_ID	instance_type_id	Nova ID for instance type ('flavor') of this instance.	Nova `compute.instance.exists` event
VM_INSTANCE_TYPE	instance_type	Name of the instance type ('flavor') of this instance **Note:** Refer to the related Product Limitations topic if the `VM_INSTANCE_TYPE` is of the format `Instance_UUID`.	Nova `compute.instance.exists` event

Table 74. Virtual Machine identifiers (continued)

Name	OpenStack Nova Compute or Keystone Equivalent	Description	Source
VM_ARCHITECTURE	architecture	The VM architecture, for example, x86, POWER®, and so on.	Nova `compute.instance.exists` event
VM_OS_TYPE	os_type	The VM operating system.	Nova `compute.instance.exists` event
VIRTIMG_REFURL	image_ref_url	Image URL, from Glance, that this instance was created from.	Nova `compute.instance.exists` event
VIRTIMG_REF	image_meta.base_image_ref	The image the instance was built from	Nova `compute.instance.exists` event

Table 75. Virtual Machine Resources - Output for each ID.

Name	OpenStack Nova Compute Equivalent	Description	Source
VM_VCPUS	vcpus	Number of Virtual CPUs allocated for this instance	Nova `compute.instance.exists` event
VM_MEMORY	memory_mb	Memory allocation for this instance (MB)	Nova `compute.instance.exists` event
VM_DISK	disk_gb	Disk allocation for this instance (GB)	Nova `compute.instance.exists` event
VM_ROOT	root_gb	Root allocation for this instance (GB)	Nova `compute.instance.exists` event
VM_EPHEMERAL	ephemeral_gb	Ephemeral allocation for this instance (GB)	Nova `compute.instance.exists` event

Related concepts:
Product limitations

Volume identifiers and resources:

This topic describes the identifier and resources defined from the OpenStack Cinder Volume notifications.

Table 76. Standard fields

Name	Volume or Keystone Equivalent	Description	Source
Start Date of Usage	timestamp/ audit_period_beginning	Date in format YYYYMMDD	Cinder `volume.exists` event
End Date of Usage	timestamp/ audit_period_ending	Date in format YYYYMMDD	Cinder `volume.exists` event

Table 76. Standard fields (continued)

Name	Volume or Keystone Equivalent	Description	Source
Start Time of Usage	timestamp/ audit_period_beginning	Time in format HH:MM:SS	Cinder `volume.exists` event
End Time of Usage	timestamp/ audit_period_ending	Time in format HH:MM:SS	Cinder `volume.exists` event

Table 77. Ownership Identifiers

Name	Volume or Keystone Equivalent	Description	Source
TENANT_ID	tenant_id	Tenant UUID	Cinder `volume.exists` event
TENANT_NAME	`{"tenants": [ {"name": "customer-x"} ] }`	Tenant name	Keystone **HTTP GET** /tenants
TENANT_DESCRIPTION	`{"tenants": [ {"description":"None"} ] }`	Tenant description	Keystone **HTTP GET** /tenants
TENANT_ENABLED	`{"tenants": [ {"enabled":true} ] }`	If Tenant is enabled or not	Keystone **HTTP GET** /tenants
USER_ID	user_id	User UUID	Nova `compute.instance.exists` event
USER_NAME	`{"users": [ {"name": "customer-x"} ] }`	User name	Keystone **HTTP GET** /users
USER_EMAIL	`{"users": [ {"email":"None"} ] }`	User email address	Keystone **HTTP GET** /users
USER_ENABLED	`{"users": [ {"enabled":true} ] }`	If user is enabled or not	Keystone **HTTP GET** /users

Table 78. Placement identifiers

Name	Volume or Keystone Equivalent	Description	Source
REGION	Derived from PUBLISHER_ID or PROVIDER_ID or / OpenStack REST volume collector	Identifier which signifies which OpenStack instance the events are generated from.	SCCM Job File

Table 78. Placement identifiers (continued)

Name	Volume or Keystone Equivalent	Description	Source
PROVIDER_ID	provider_id	OpenStack Broker Identifier	MCS provider configuration file.
PUBLISHER_ID	publisher_id	Identifier which signifies which AMQP broker (OpenStack) instance the events are generated from	Cinder `volume.exists` event

Table 79. History identifiers

Name	Volume or Keystone Equivalent	Description	Source
VOL_CREATED_AT	created_at	Timestamp for when this instance's record was created in Cinder (YYYY-MM-DD hh:mm:ss)	Cinder `volume.exists` event
VOL_LAUNCHED_AT	launched_at	Timestamp for when this instance was last launched (YYYY-MM-DD hh:mm:ss)	Cinder `volume.exists` event

Table 80. Volume identifiers

Name	Volume or Keystone Equivalent	Description	Source
VOL_ID	volume_id	Cinder instance ID of this instance	Cinder `volume.exists` event
VOL_NAME	display_name	User selected display name for instance	Cinder `volume.exists` event
VOL_STATUS	status	Current state of instance, such as **`active`** or **`deleted`**)	Cinder `volume.exists` event
VOL_TYPE	volume_type	Volume type	Cinder `volume.exists` event

Table 81. Volume Resources - Output for each VOL_ID

Name	Volume or Keystone Equivalent	Description	Source
VOL_SIZE	size	Size of disk allocation for this volume (GB)	Cinder `volume.exists` event

OpenStack job file

As part of the installation of SmartCloud Cost Management, sample job files are added to the sample job files directory.

The following job files are used to process OpenStack utilization data for the current OpenStack release:

Context job file:

As part of the automatic configuration of SmartCloud Cost Management, a job file called `OpenStackContext.xml` is added to the job files directory. For more information about the post install configuration, see the related topic.

The `OpenStackContext.xml` job file is used to process the OpenStack Keystone Context data for the current OpenStack release. When the `OpenStackContext.xml` job file is run, it executes the required active Keystone job.

The following topic describes the job contained in the `OpenStackContext.xml` job file.

OpenStackKeystoneContext job:

The `OpenStackKeystoneContext` job is used to collect domain, project and user information from the Keystone Identity Service API. This data is used to provide information such as domain, project, and user names to the utilization data collected.

The domain, project, and user details that are produced by the collector are loaded into the SmartCloud Cost Management database as conversion mappings. These conversion mappings are then used to map meaningful context identifiers into the utilization data. The collector data also produces account code security related identifiers based on user roles in Keystone. These identifiers can then feed into the `CreateAccountRelationship` and `CreateUserRelationship` stages creating users, user groups, and clients in the SmartCloud Cost Management database to facilitate account code security.

The Keystone template in the `StandardTemplates.xml` file that is located in `<SCCM_install_dir>/collectors/KEYSTONE` defines what metrics are collected from both the domain, project, and user Keystone resources. The `StandardTemplates.xml` file also defines what CSR identifier names are used to map the collected metrics to. For more information about the OpenStack identifiers and resources, see the related reference topic.

This data collector uses the Integrator program to convert data collected by the collector into a CSR or CSR+ file. For more information about the Integrator job file structure, see the Reference guide.

The `OpenStackKeystoneContext` job contains the following parameters that are specific to Keystone collection:

```
<Input name="CollectorInput" active="true">
              <Collector name="KEYSTONE">
                <WebService dataSourceName="os_keystone" />
                <MappingTemplate filePath="%HomePath%/collectors/KEYSTONE/StandardTemplates.xml" name
              </Collector>
              <Parameters>
              <Parameter name="resourcePath" value="/users, /users/{keystoneUserId}/projects, /domain
              <Parameter name="chainedResourceTemplateKeys" value="KEYSTONE_USER, KEYSTONE_PROJECT,
```

```
<Parameter name="tenant" value="admin" DataType="String" />
<Parameter name="domain" value="Default" DataType="String" />
<Parameter name="openStackCompatibleAccountCodeStructure" value="STANDARD" DataType=
<Parameter name="adminUserGroup" value="Admin" DataType="String" />
</Parameters>
```

These parameters are described in the following table:

Parameter	Description/Values
WebService-dateSourceName	The name of the Web Service SmartCloud Cost Management data source. The data source points to the base URL of the Keystone V3 Identity Service API and contains the authentication credentials. For more information about setting up the Web Service data source, see the related configuration topic.
MappingTemplate-filePath	The path of the template mapping file. You can generate custom template files and use them instead of the standard templates defined in `<SCCM_install_dir>/collectors/KEYSTONE/StandardTemplates.xml` by specifying a different file name.
MappingTemplate-name	The name of the template. This value controls the type of records which are read from Keystone and processed by the `OpenStackKeystoneContext` job. Valid values as contained in the default Keystone `StandardTemplates.xml` are: • **KEYSTONE** – used for the combined domain, project, user, and account security information. • **KEYSTONE_PROJECT** – used for the project information. • **KEYSTONE_USER** – used for the user information.

Parameter	Description/Values
resourcePath	The **resourcePath** parameter is set to the base URLs of the resources that are used when collecting from Keystone. The **resourcePath** must point to a legal Keystone API resource such as `/tenants` or `/users` and must begin with a `"/"`. In the sample, multiple URLs are specified as we want to join our resource API calls together to collect data from multiple related resources. Therefore some results from one API call are used in the subsequent API calls and so on. • **/users** is specified as the first resource as we want to collect user information. • **/users/{keystoneUserId}/projects** is specified as the second resource as we want to collect project information relating the user. The dynamic **{keystoneUserId}** parameter is populated from the result of the **/users** resource call and is defined as the name of `InputField` in the Keystone template in `StandardTemplate.xml`. • **/domains/{keystoneProjectDomainId}** is specified as the third resource as we want to collect domain information. The dynamic **{keystoneProjectDomainId}** parameter is populated from the result of the previous resource call. The **resourcePath** is identified as being chained, as it contains multiple resources with the attribute `format="chained"`
chainedResourceTemplateKeys	The **chainedResourceTemplateKeys** parameter is set because the **resourcePath** parameter is chained with multiple resources. The **chainedResourceTemplateKeys** must contain the same number of fields as the number of resources contained in the chained **resourcePath**, in this case 3. The 3 **chainedResourceTemplateKeys**, **KEYSTONE_USER**, **KEYSTONE_PROJECT**, and **KEYSTONE_DOMAIN** indicate which **InputFields** in the Keystone template in `StandardTemplate.xml` relate to which resource APIs. For example, the **InputFields** with **key="KEYSTONE_USER"** relate to data returned from the **/users** resource.
tenant	The name of a tenant (Project) is required to authenticate with the Keystone Identity Service API. The tenant is used to request a security token that is used in all subsequent REST calls that must be made to the APIs.
domain	The name of a domain, which the tenant/Project is part of, is required to authenticate with the Keystone Identity Service API. The domain is used to request a security token that is used in all subsequent REST calls. These REST calls must be made to the APIs.

Parameter	Description/Values
openStackCompatibleAccountCodeStructure	The **openStackCompatibleAccountCodeStructure** parameter is set to the name of the Account Code Structure that is used when generating any of the account code security related identifiers. These identifiers are used to feed into the `CreateAccountRelationship` and `CreateUserRelationship` stages so users, user groups, and clients are created automatically. This is an optional stage and if it is not specified, the account code security related identifiers are not generated.
adminUserGroup	The **adminUserGroup** parameter is set to the name of the SmartCloud Cost Management user group that is used to associate Keystone Cloud Admins with.

The `OpenStackKeystoneContext` job also contains 3 `UpdateConversionFromRecord` stages that are used to load conversion mappings from the resources collected. These conversion mappings are used by the following job files:

- `OpenStackVMInstances.xml` - VM Instances job file
- `OpenStackVolumes.xml` - Volumes job file
- `OpenStackImages.xml` - Images job file

The conversion mappings are used by these job files to determine the user, project, and domain names when a user or project ID is found in the record that is being processed. See the related topic for more information about these job files.

Note: The **allowConversionEntryUpdate** parameter is set to **none** by default. This parameter must not be changed as the project and user name is used in the IBM Cloud Orchestrator account code. As a result, any updates to the initial name value in Keystone must not be applied to the conversion mapping. Changing the value to `forwardOnly` would result in resources being tracked against a different account code after the values were changed.
The `OpenStackContext.xml` job file contains the following parameters that are specific to the `UpdateConversionFromRecord` stage. For more information on this stage, see the Reference guide. This stage is used to load conversion mappings from the resources that are collected from the tenants and users Keystone Identity Service API.

```
<Stage name="UpdateConversionFromRecord" active="true">
 <Identifiers>
  <Identifier name="KEYSTONE_USER_NAME">
   <FromIdentifiers>
    <FromIdentifier name="KEYSTONE_USER_ID" />
   </FromIdentifiers>
  </Identifier>
 </Identifiers>
 <Files>
 <File name="%CollectorLogs%/Keystone/KeystoneUsersConversionException.txt" type="exception" format="CSROutput" />
 </Files>
 <Parameters>
  <Parameter process="KeystoneUsers" />
  <Parameter useUsageEndDate="false" />
  <Parameter allowNewConversionEntries="true" />
  <Parameter allowConversionEntryUpdate="none" />
  <Parameter strictLockdown="true" />
  <Parameter exceptionprocess="true" />
 </Parameters>
</Stage>
```

The `OpenStackKeystoneContext` job also contains 2 stages related to facilitating account code security. The `CreateAccountRelationship` and

CreateUserRelationship stages take the security related identifiers generated by the Keystone collector so users, user groups, and clients are created automatically. For more information on these stage see the related topics in the Reference guide.

VM Instances job file:

As part of the installation of SmartCloud Cost Management, a job file called OpenStackVMInstances.xml is added to the job files directory.

The OpenStackVMInstances.xml job file is used to process the OpenStack Nova Compute utilization data for the current OpenStack release and it contains both active and inactive jobs. When the OpenStackVMInstances.xml job file is run, it executes the required active jobs in the sequence displayed below:

- OpenStackNovaComputeScan
- OpenStackNovaComputeProess
- OpenStackNovaComputeTemplateMappingSample1
- OpenStackNovaComputeBillAndLoad

The job file also contains inactive jobs that are not required, but can be made active depending on the pricing structure you require. These inactive jobs are:

- OpenStackNovaComputeTemplateMappingSample2
- OpenStackNovaComputeTemplateMappingSample3

Note: The inactive jobs must be activated before running the job file and only one inactive job can be made active at any one time. The topics in this section describe each of the active and inactive jobs contained in the OpenStackVMInstances.xml job file.

OpenStackNovaComputeScan job:

The OpenStackNovaComputeScan job is used to scan all the Nova Compute utilization data files for a specific time period. The data returned is then concatenated into one file that is used as input for the OpenStackNovaComputeProcess job.

The Nova Compute utilization data files that are scanned are located in the <CollectorLogs>/nova_compute/.. directory. The Metering Control Service (MCS) is configured to write logs for each Nova Compute utilization data file to subdirectories under the nova_compute folder.

OpenStackNovaComputeProcess job:

The OpenStackNovaComputeProcess job is used to process OpenStack Nova compute utilization data.

The OpenStackNovaComputeProcess job defines a **nova_compute** process that contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS) collector.
2. Creates a **DOMAIN_NAME** identifier from the **TENANT_ID** identifier that is found in the Nova Compute CSR file. This **DOMAIN_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The **DOMAIN_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic.

3. Creates a **PROJECT_NAME** identifier from the **TENANT_ID** identifier that is found in the Nova Compute CSR file. This **PROJECT_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The **PROJECT_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic.
4. Creates a **USER_NAME** identifier from the **USER_ID** identifier that is found in the Nova Compute CSR file. The **USER_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic. This **USER_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The following code sample shows how the `CreateIdentifierFromTable` stage is used to create the **USER_NAME** identifier:

```
<!-- STAGE: Create USER_NAME collected from Keystone from USER_ID -->
      <Stage name="CreateIdentifierFromTable" active="true">
         <Identifiers>
            <Identifier name="USER_NAME">
              <FromIdentifiers>
                <FromIdentifier name="USER_ID"/>
              </FromIdentifiers>
            </Identifier>
         </Identifiers>
         <DBLookups>
            <!-- last discriminator is default, job id used as default, cacheSize of 24 is def
            <DBLookup discriminator="last" process="KeystoneUser" cacheSize="24"/>
         </DBLookups>
```

5. Creates a **REGION** identifier from either the **PROVIDER_ID** or the **PUBLISHER_ID** identifier. Depending on your OpenStack setup, you must enable the step for one of these identifiers. The **PROVIDER_ID** is enabled to create the **REGION** identifier by default.
6. Creates the default IBM Cloud Orchestrator account code with the following identifiers as levels:
   - **DOMAIN_NAME**
   - **PROJECT_NAME**
   - **USER_NAME**
   - **VM_NAME**

OpenStackNovaComputeTemplateMappingSample1 job:

The `OpenStackNovaComputeTemplateMappingSample1` job is used to map the Nova resources to the rate codes defined in the Virtual Systems Rate template.

The `OpenStackNovaComputeTemplateMappingSample1` job contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS).
2. Renames the Nova resource fields to the rate codes defined by the `Virtual Systems` Rate template.
3. Converts the **VSVMSTORRS** from Megabytes to Gigabytes.
4. Converts **VSVMMEMORY** from Megabytes to Gigabytes.
5. Writes the mapped output to **%LogDate_End%-ICO.txt**.

OpenStackNovaComputeTemplateMappingSample2 job:

The NovaComputeTemplateMappingSample2 job is an optional job and it is used as an example of how to map the Nova resources to the rate codes defined in the License Charges, Infrastructure Charges, and the Hosting Charges rate templates. This job must be activated if you want to use this style of pricing model when processing and billing the OpenStack data.

Note: If you intend to use the OpenStackNovaComputeTemplateMappingSample2 job, you must ensure that the **inputFile** parameter in the OpenStackNovaComputeBillAndLoad job is updated to **%LogDate_End%-ICO.txt**. For more information about this, see the related reference topic about the OpenStackNovaComputeBillAndLoad job.

The OpenStackNovaComputeTemplateMappingSample2 job contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS).
2. The first rate code that is created is a License Charges rate template rate code. The value of the rate code in this step is determined by the **VM_OS_TYPE** identifier. However, other licensing charges can also be applied to an identifier, for example, the **PATTERN_NAME**. The licensing charges rate code is built by using a combination of stages that create and append intermediary identifiers and resources together. It also uses a conversion table to map the **VM_OS_TYPE** identifier value to a rate dimension element short code, for example, **OS_RATEDIMENSION_ELEMENT**. You can update the sample License Charges rate template and conversion table so that any additional values that are generated in the CSR records from OpenStack can be charged for.
3. The second set of rate codes that are created are the Infrastructure Charges rate template rate codes. The value of the rate codes in this step is determined by combining the base Nova resources with the value of the **VM_ARCHITECTURE** identifier. The Infrastructure Charges rate codes are built using a combination of stages that create and append intermediary identifiers together. It also uses a conversion table to map the **VM_ARCHITECTURE** identifier value to a rate dimension element short code, for example, **OS_RATEDIMENSION_ELEMENT**. A number of **RenameResource** stages are used to map the final Infrastructure Charges rate template rate codes.
4. The third rate code type created is a Hosting Charges rate template rate code. The value of the rate code in this step is determined by a combination of the **AVAILABILITY_ZONE** and the **VM_ARCHITECTURE** identifiers. The Hosting Charges rate template rate code is built using a combination of stages that create and append intermediary identifiers and resource together. It also uses two conversion tables to map the **AVAILABILITY_ZONE** and the **VM_ARCHITECTURE** identifier values to their rate dimension element short codes, for example **OS_RATEDIMENSION_ELEMENT**. This sample Hosting Charges rate template and conversion table can be updated to include any additional values that are generated in the CSR records from OpenStack, that the end user would like to charge for.
5. Writes the mapped output to **%LogDate_End%-ICO.txt**.

OpenStackNovaComputeTemplateMappingSample3 job:

The OpenStackNovaComputeTemplateMappingSample3 job is an optional job and it is used as an example of how to map the Nova resources to rate codes that are defined in the Hosting Charges with VM Sizes and License Charges rate templates. This job must be activated if you want to use this style of pricing model when processing and billing the OpenStack data.

Note: If you intend to use the OpenStackNovaComputeTemplateMappingSample3 job, you must ensure that the **inputFile** parameter in the OpenStackNovaComputeBillAndLoad job is updated to **%LogDate_End%-ICO.txt**. For more information about this, see the related reference topic about the OpenStackNovaComputeBillAndLoad job.

The OpenStackNovaComputeTemplateMappingSample3 job contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS).
2. The first rate code that is created is a License Charges rate template rate code. The value of the rate code in this step is determined by the **VM_OS_TYPE** identifier. However, other licensing charges can also be applied to an identifier, for example, the **PATTERN_NAME**. The licensing charges rate code is built by using a combination of stages that create and append intermediary identifiers and resources together. It also uses a conversion table to map the **VM_OS_TYPE** identifier value to a rate dimension element short code, for example, **OS_RATEDIMENSION_ELEMENT**. The sample License Charges rate template and conversion table can be updated to include any additional values that come through in the feed that you want to rate.
3. The second rate code that is created is a Hosting Charges with VM Sizes rate template rate code. The difference between the pricing model demonstrated in the Hosting Charges with VM Sizes rate template and the OpenStackNovaComputeTemplateMappingSample2 job is that the charge is no longer attributed to the infrastructure resources collected by the MCS, for example, CPU, Memory and Storage. Instead the charge is associated with the overall size of the VM that is defined in the **VM_INSTANCE_TYPE** identifier, for example tiny, small, large, and so on. As a result of this, the value of the rate codes in this step is determined by a combination of the **AVAILABILITY_ZONE**, the **VM_ARCHITECTURE**, and the **VM_INSTANCE_TYPE** identifiers. The Hosting Charges with VM Sizes rate template rate code is built using a combination of stages that create and append intermediary identifiers and resources together. It also uses a conversion table to map the **AVAILABILITY_ZONE**,**VM_ARCHITECTURE**, and **VM_INSTANCE_TYPE** identifier values to their rate dimension element short code, for example, **OS_RATEDIMENSION_ELEMENT**. The Hosting Charges with VM Sizes rate template and conversion table can be updated to include any additional values that come through in the feed that you want to rate.
4. Write the mapped output to **%LogDate_End%-ICO.txt**.

OpenStackNovaComputeBillandLoad job:

The OpenStackNovaComputeBillandLoad job is used to bill all of the data that has been processed by the previous jobs. It loads the data that has been billed into the SmartCloud Cost Management database where it is used for reporting.

Volumes job file:

As part of the installation of SmartCloud Cost Management, a job file called OpenStackVolumes.xml is added to the job files directory.

The OpenStackVolumes.xml job file is used to process the OpenStack Cinder Volume utilization data and it contains active jobs. When the OpenStackVolumes.xml job file is run, it executes the required active jobs in the sequence displayed below:

- OpenStackCinderVolumeScan
- OpenStackCinderVolumeProess
- OpenStackCinderVolumeTemplateMappingSample1
- OpenStackCinderVolumeBillAndLoad

Note: The topics in this section describe each of the active jobs contained in the OpenStackVolumes.xml job file.

OpenStackCinderVolumeScan job:

The OpenStackCinderVolumeScan job is used to scan all the Cinder Volume utilization data files for a specific time period. The data returned is then concatenated into one file that is used as input for the OpenStackCinderVolumeProcess job.

The Cinder Volume utilization data files that are scanned are located in the <CollectorLogs>/cinder_volume/.. directory. The Metering Control Service (MCS) is configured to write logs for each Cinder Volume utilization data file to subdirectories under the cinder_volume folder.

OpenStackCinderVolumeProcess job:

The OpenStackCinderVolumeProcess job is used process OpenStack Cinder Volume utilization data.

The OpenStackCinderVolumeProcess defines a cinder_volume process that contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS) collector.
2. Creates a **DOMAIN_NAME** identifier from the **TENANT_ID** identifier that is found in the Cinder Volume CSR file. This **DOMAIN_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The **DOMAIN_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic.
3. Creates a **PROJECT_NAME** identifier from the **TENANT_ID** identifier that is found in the Cinder Volume CSR file. This **PROJECT_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The **PROJECT_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic.

4. Creates a **USER_NAME** identifier from the **USER_ID** identifier that is found in the Cinder Volume CSR file. The **USER_NAME** conversion is supplied by the Keystone collector. For more information, see the related topic. This **USER_NAME** identifier is required as it is part of the default IBM Cloud Orchestrator account code. The following code sample shows how the `CreateIdentifierFromTable` stage is used to create the **USER_NAME** identifier:

```
<!-- STAGE: Create USER_NAME collected from Keystone from USER_ID -->
      <Stage name="CreateIdentifierFromTable" active="true">
         <Identifiers>
            <Identifier name="USER_NAME">
              <FromIdentifiers>
                <FromIdentifier name="USER_ID"/>
              </FromIdentifiers>
            </Identifier>
         </Identifiers>
         <DBLookups>
            <!-- last discriminator is default, job id used as default, cacheSize of 24 is def
            <DBLookup discriminator="last" process="KeystoneUser" cacheSize="24"/>
         </DBLookups>
```

5. Creates a **REGION** identifier from either the **PROVIDER_ID** or the **PUBLISHER_ID** identifier. Depending on your OpenStack set up, you must enable the step for one of these identifiers. The **PROVIDER_ID** is enabled to create the **REGION** identifier by default.
6. Creates the default IBM Cloud Orchestrator account code with the following identifiers as levels:
   - **DOMAIN_NAME**
   - **PROJECT_NAME**
   - **USER_NAME**
   - **VOL_NAME**

OpenStackCinderVolumeTemplateMappingSample1 job:

The `OpenStackCinderVolumeTemplateMappingSample1` job is used to map the Cinder resources to the rate codes defined in the Virtual Systems rate template.

The `OpenStackCinderVolumeTemplateMappingSample1` job contains the following steps:

1. Reads the CSR file that is generated by the Metering Control Service (MCS).
2. Renames the Cinder resource fields to the rate codes defined by the Virtual Systems rate template.
3. Writes the mapped output to **%LogDate_End%-ICO.txt**.

OpenStackCinderVolumeBillandLoad job:

The `OpenStackCinderVolumeBillandLoad` job is used to bill all of the data that has been processed by the previous jobs. It loads the data that has been billed into the SmartCloud Cost Management database where it is used for reporting.

IBM PowerVM HMC data collector

The IBM Hardware Management Console (HMC) collector collects data from IBM HMC version 7.2.0.

Configuring HMC to enable SmartCloud Cost Management HMC collection

This topic describes how to configure HMC to enable SmartCloud Cost Management HMC collection.

Procedure

1. The SmartCloud Cost Management IBM Hardware Management Console (HMC) collector uses Secure Shell (SSH) to gather metrics from the HMC. The PowerVM® servers managed by the HMC need to be configured to produce the data for collection. For HMC version 7.2.0, the **'chlparutil'** command can be used for the configuration. Refer to the IBM HMC documentation for more details.
2. Sample rates of 0, 30 (30 seconds), 60 (60 seconds), 300 (5 minutes), 1800 (30 minutes), and 3600 (1 hour) are supported by the HMC for sample rate configuration. This is the rate, in seconds, that is used to sample the data i.e. the data granularity. For usage and accounting purposes, a sampling rate of 3600 seconds (1 hour) is recommended. Refer to the IBM HMC documentation for more details.

   Note: A managed system can be managed by one or more HMCs. Duplicate data will be generated for a system managed by different HMCs. This should be taken into consideration if collecting data from HMCs with this infrastructure.

HMC collector log file format

This topic describes the record fields in the log file produced by the IBM® HMC collector.

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

- LPAR Sample records, which provide usage data HMC sample events for partitions.
- Memory pool Sample records, which provide usage data HMC sample events for shared memory pools.
- Processor Pool Sample records, which provide usage data HMC sample events for shared processor pools.
- System Sample records, which provide usage data HMC sample events for managed systems.

Table 82. Log File Format – LPAR Sample Records

Field Name	Description/Values
recordtype	SmartCloud Cost Management record type identifier. Value is TUAMHMC.
IntervalStartDateTime	Start collection interval.
IntervalEndDateTime	End collection interval.
HMCHostName	HMC host name or IP address.
SystemName	User defined name of the managed system.
SerialNum	Managed system type, model and serial number.

Table 82. Log File Format – LPAR Sample Records (continued)

Field Name	Description/Values
Rate	The sample rate which specifies the granularity of the record data.
ProcCyclesPerSec	Processing cycles per second on one physical processor. This value is static for a particular managed system.
State	This is the state of the LPAR at the time the event was collected. Available for hourly rate only.
StartDateTime	The time on the HMC that the event was collected.
EventType	The type of event. Value is **sample**.
ResourceType	The type of system resource for which the event was collected. Value is **lpar**.
SysDateTime	The time on the managed system that the sample was taken.
LPARName	The user-defined name of the partition at the time the event was collected.
LPARId	The unique integer identifier for the partition.
SharedProcPoolName	The user-defined name of the shared processor pool, at the time the event was collected, that the partition is in.
SharedProcPoolId	The unique integer identifier for the shared processor pool that the partition is in.
ProcMode	The processing mode for the partition. Possible values are **ded** or **shared**.
NumProcs	The number of processors or virtual processors assigned to the partition.
EntitledCycles	The number of processing cycles to which the partition has been entitled since the managed system was started. This value is based on the number of processing units assigned to the partition, and may be greater than or smaller than the number of cycles actually used.
CappedCycles	The number of capped processing cycles utilized by this partition since the managed system was started.
UncappedCycles	The number of uncapped processing cycles utilized by this partition since the managed system was started.
CurrentMemory	The amount of memory (in megabytes) assigned to the partition. For shared memory partitions, this is the amount of logical memory assigned to the partition.
SharingMode	The sharing mode of the partition. Possible values are *keep_idle_procs*, *share_idle_procs*, *share_idle_procs_active*, *share_idle_procs_always*, *cap*,or *uncap*.
5250CPWPercent	The *5250* CPW percent assigned to the partition.
ProcUnits	The number of processing units assigned to the partition.

Table 82. Log File Format – LPAR Sample Records (continued)

Field Name	Description/Values
UncappedWeight	The current weighted average of processing priority when in uncapped sharing mode. The smaller the value, the lower the weight. Possible values are *0 - 255*.
MemMode	The memory mode for the partition. Possible values are **ded** or **shared**.
TimeCycles	The number of time cycles since the managed system was started.
IOEntitledMem	The amount of I/O entitled memory (in megabytes) assigned to the shared memory partition.
MapIOEntMem	The amount of I/O entitled memory (in megabytes) currently mapped by the shared memory partition.
PhyRunMem	The runtime amount of physical memory (in megabytes) allocated to the shared memory partition.
RunMemWeight	The runtime relative memory priority for the shared memory partition. The smaller the value, the lower the priority. Possible values are *0 - 255*.
MemOvgCoop	The difference between the shared memory partition's assigned memory overcommitment and its actual overcommitment. A positive value means the partition is using less memory than system firmware has requested it to use.
SharedCycActive	The number of dedicated processing cycles shared by this partition while it has been active since the managed system was started.

Table 83. HMC Log File Format – Memory Pool Sample Records

Field Name	Description/Values
recordtype	SmartCloud Cost Management record type identifier. Value is TUAMHMC.
IntervalStartDateTime	Start collection interval.
IntervalEndDateTime	End collection interval.
HMCHostName	HMC host name or IP address.
SystemName	User defined name of the managed system.
SerialNum	Managed system type, model and serial number.
Rate	The sample rate which specifies the granularity of the record data.
StartDateTime	The time on the HMC that the event was collected.
EventType	The type of event. Value is **sample**.
ResourceType	The type of system resource for which the event was collected. Value is **mempool**.
SysDateTime	The time on the managed system that the sample was taken.
ProcCyclesPerSec	Processing cycles per second on one physical processor. This value is static for a particular managed system.
PoolMem	The size (in megabytes) of the shared memory pool.

Table 83. HMC Log File Format – Memory Pool Sample Records (continued)

Field Name	Description/Values
LPARRunMem	The total amount of logical memory (in megabytes) assigned to all of the partitions in the shared memory pool.
LPARIOEntMem	The total amount of I/O entitled memory (in megabytes) assigned to all of the partitions in the shared memory pool.
LPARIOMapEntMem	The total amount of I/O entitled memory (in megabytes) currently mapped by all of the partitions in the shared memory pool.
SysFirmPoolMem	Amount of memory, in megabytes, in the shared memory pool that is being used by system firmware.
PageFaults	The total number of page faults that have occurred since the shared memory pool was created.
PageDelay	The total page-in delay, in microseconds, spent waiting for page faults since the shared memory pool was created.

Table 84. HMC Log File Format – Processor Pool Sample Records

Field Name	Description/Values
recordtype	SmartCloud Cost Management record type identifier. Value is TUAMHMC.
IntervalStartDateTime	Start collection interval.
IntervalEndDateTime	End collection interval.
HMCHostName	HMC host name or IP address.
SystemName	User defined name of the managed system.
SerialNum	Managed system type, model and serial number.
Rate	The sample rate which specifies the granularity of the record data.
StartDateTime	The time on the HMC that the event was collected.
EventType	The type of event. Value is **sample**.
ResourceType	The type of system resource for which the event was collected. Value is **procpool**.
SysDateTime	The time on the managed system that the sample was taken.
ProcCyclesPerSec	Processing cycles per second on one physical processor. This value is static for a particular managed system.
SharedProcPoolName	The user-defined name of the shared processor pool at the time the event was collected.
ShraedProcPoolId	The unique integer identifier for the shared processor pool.
TimeCycles	The number of time cycles since the managed system was started.
TotalPoolCycles	The total number of processing cycles available in the physical processor pool or shared processor pool since the managed system was started.

Table 84. HMC Log File Format – Processor Pool Sample Records (continued)

Field Name	Description/Values
UtilizedPoolCycles	The number of processing cycles in the physical processor pool or shared processor pool that have been utilized since the managed system was started.

Table 85. HMC Log File Format – System Sample Records

Field Name	Description/Values
recordtype	SmartCloud Cost Management record type identifier. Value is TUAMHMC.
IntervalStartDateTime	Start collection interval.
IntervalEndDateTime	End collection interval.
HMCHostName	HMC host name or IP address.
SystemName	User defined name of the managed system.
SerialNum	Managed system type, model and serial number.
Rate	The sample rate which specifies the granularity of the record data.
StartDateTime	The time on the HMC that the event was collected.
EventType	The type of event. Value is **sample**.
ResourceType	The type of system resource for which the event was collected. Value is **sys**.
SysDateTime	The time on the managed system that the sample was taken.
ProcCyclesPerSec	Processing cycles per second on one physical processor. This value is static for a particular managed system.
State	This is the state of the managed system at the time the event was collected.
SysProcUnits	The number of configurable system processing units.
SysMemory	The amount of configurable system memory (in megabytes).
AvailSysProcUnits	The number of processing units available to be assigned to partitions.
Avail5250CPWPercent	The *5250* CPW percent available to be assigned to partitions.
AvailSysMem	The amount of memory (in megabytes) available to be assigned to partitions.
SysFirmMem	Amount of memory, in megabytes, on the managed system that is being used by system firmware.
ProcCyclesPerSec	Processing cycles per second on one physical processor. This value is static for a particular managed system.

Refer to HMC Data Commands topic for more details on the commands run and attributes gathered on the HMC.

Identifiers and resources defined by HMC collector

The HMC data collector defines the identifiers and resources described in this topic.

By default, the data collected by the HMC collector is defined as chargeback identifiers and resource rate codes in the `${TUAM_HOME}/collectors/hmc/StandardTemplate.xml`.

Table 86. Default HMC Identifiers and Resources – LPAR Sample

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers		
`HMCHostName`	HMC host name or IP address.	–
`SystemName`	User defined name of the managed system.	–
`SerialNum`	Managed system type, model and serial number.	–
`LPARName`	User defined name of the partition.	–
`LPARId`	Unique identifier of the partition.	–
`LPARState`	State of the LPAR.	–
`SharedProcPoolName`	User defined name of the shared processor pool.	–
`SharedProcPoolId`	Unique identifier of the shared processor pool.	–
`ProcMode`	Partition processing mode.	–
`EventType`	The type of event. Value is **sample**.	–
`ResourceType`	The type of system resource for which the event was collected. Value is **lpar**.	–
Resources		
`NumProcs`	Number of processors/virtual processors assigned to a partition.	LPNUMCPU
`EntitledCycles`	Processing cycles a partition is entitled to (Mega-Cycles).	LPENTCAP
`CappedCycles`	Capped processing cycles used by a partition (Mega-Cycles).	LPCAPCYC
`UncappedCycles`	Uncapped processing cycles used by a partition (Mega-Cycles).	LPUNCAP
`CurrentMemory`	Memory assigned to a partition (MB).	LPCURMEM
`EntitledHours`	CPU hours that a partition is entitled to (hours).	LPENTCPH

Table 86. Default HMC Identifiers and Resources – LPAR Sample (continued)

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
CappedHours	Capped CPU hours used by a partition (hours).	LPCAPH
UncappedHours	Uncapped CPU hours used by a partition (hours).	LPUNCAPH

Table 87. Default HMC Identifiers and Resources – Memory Pool Sample

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers		
HMCHostName	HMC host name or IP address.	–
SystemName	User defined name of the managed system.	–
SerialNum	Managed system type, model and serial number.	–
EventType	The type of event. Value is **sample**.	–
ResourceType	The type of system resource for which the event was collected. Value is **mempool**.	–
Resources		
PoolMem	Size of shared memory pool (MB).	SMPCUMEM

Table 88. Default HMC Identifiers and Resources – Processor Pool Sample

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
HMCHostName	HMC host name or IP address.	–
SystemName	User defined name of the managed system.	–
SerialNum	Managed system type, model and serial number.	–
SharedProcPoolName	User defined name of the shared processor pool.	–
SharedProcPoolId	Unique identifier of the shared processor pool.	–
EventType	The type of event. Value is **sample**.	–
ResourceType	The type of system resource for which the event was collected. Value is **procpool**.	–
Resources		

Table 88. Default HMC Identifiers and Resources – Processor Pool Sample (continued)

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
TotalPoolCycles	Processing cycles available to a shared processor pool (Mega-Cycles).	SPPTLCYC
UtilizedPoolCycles	Processing cycles used by a shared processor pool (Mega-Cycles).	SPPULCYC
TotalPoolHours	CPU hours available to a shared processor pool (hours).	SPPTLH
UtilizedPoolHours	CPU hours used by a shared processor pool (hours).	SPPULH

Table 89. Default HMC Identifiers and Resources – System Sample

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
HMCHostName	HMC host name or IP address.	–
SystemName	User defined name of the managed system.	–
SerialNum	Managed system type, model and serial number.	–
EventType	The type of event. Value is **sample**.	–
ResourceType	The type of system resource for which the event was collected. Value is **sys**.	–
Resources		
SysProcUnits	Configurable system processing unit.	SYSCGCPU
SysMemory	Configurable system memory (MB).	SYSCGMEM
AvailSysProcUnits	Available processing units for partitions.	SYSAVCPU
AvailSysMem	Available memory partitions (MB).	SYSAVMEM

Setting up SmartCloud Cost Management for HMC data collection

The sections that follow provide details about setting up SmartCloud Cost Management for HMC data collection.

Creating a Server Connection data source

You must create a data source in Administration Console that points to the HMC. The data source is referenced in the HMC collector job runner file in the Administration Console:

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Server** as the **Data Source Type**.
2. Click **Create Data Source**.
3. Complete the following:

   Data Source Name
   Type the name that you want to assign to the data source.

   Note: The following are invalid characters for a data source name: "/", "\","","":","?","<",">",".","|",".".

   Username
   Type the HMC Username.

   Password
   Type the HMC password.

   Host Type the host name, IP address, or IP name of the HMC. If you are using an Internet Protocol Version 6 (IPv6) address as the hostname, the IP address must be specified as follows:

   - Enclose the address with square brackets. For example, IPv6 address `aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb` should be specified as `[aaaa:bbbb:cccc:dddd:eeee:ffff:aaaa:bbbb]`.

   Protocol
   Enter SSH for HMC. If this field is left empty, it defaults to SSH.

   Port Enter the port number. If this field is left empty, it defaults to 22.

   Timeout
   Enter the timeout value. If this field is left empty, it defaults to 180000.

   Private Key File
   The private key file is a file which contains an encrypted key for authenticating during a secure connection. The passphrase is used to encrypt or decrypt the key file. Enter a valid path to the key file.

   Restricted Shell
   Select this check box for HMC.
4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the database is verified. You should see a message at the top of the screen indicating that the connection was successful.

Editing the sample job file

The HMC data collector is divided into two parts:

- A collector for the collection of data from the HMC and the writing of this data to a file per managed system of the HMC.
- A delimited collector to parse the data file(s) from the HMC into SmartCloud Cost Management for processing.

The collector uses the Integrator program to collect the data and to convert data collected by the collector into a CSR or CSR+ file. The Integrator program is run from a job file and uses the common XML architecture used for all data collection in addition to elements that are specific to Integrator.

SmartCloud Cost Management includes a sample job file, `SampleHMC.xml` that you can modify and use to process the HMC data. After you have modified and optionally renamed the `SampleHMC.xml` file, move the file to the `SCCM_install_dir>\jobfiles` directory.

The `SampleHMC.xml` file contains the following parameters that are specific to HMC collection.

```
<Input name="CollectorInput" active="true">
 <Collector name="HMC">
  <ServerConnection dataSourceName="HMCCollector"/>
  <HMCOption event="h"procpoolSample="time,event_type,resource_type,
   sys_time, shared_proc_pool_name,shared_proc_pool_id,time_cycles,total_pool_cycles,
   utilized_pool_cycles"  startTime="10:00"/>
 </Collector>
 <Parameters>
  <Parameter name="logdate" value="%LogDate_End%" DataType="DateTime"
   format="yyyyMMdd"/>
  <Parameter name="outputFolder" value="%CollectorFolder%/hmc"  DataType="String" />
  <Parameter name="timeout" value="180000" DataType="Integer"/>
 </Parameters>
 <Files>
  <File name="Exception.txt" type="exception"/>
 </Files>
</Input>
```

These parameters are described in the following table.

Table 90. HMC Parameters

Parameter	Description/Values
ServerConnection-dateSourceName	The name of the SmartCloud Cost Management data source that points to the HMC.
HMCOption-rate	This is the rate in seconds at which the HMC was configured for data granularity at the time the current data being collected was generated.
HMCOption-event	This is the HMC sampling events to be gathered. The values are as follows: • h: hourly sampling events • d: daily sampling events
HMCOption-lparSample	A delimited separated list (comma by default) of the attribute names for the required attribute values to be gathered for the lpar sample events. If not specified a default list of all lpar sample attributes as per HMC version 7.2.0 are gathered.
HMCOption-sysSample	A delimited separated list (comma by default) of the attribute names for the required attribute values to be gathered for the sys sample events. If not specified a default list of all sys sample attributes as per HMC version 7.2.0 are gathered.

Table 90. HMC Parameters (continued)

Parameter	Description/Values
HMCOption-mempoolSample	A delimited separated list (comma by default) of the attribute names for the required attribute values to be gathered for the mempool sample events. If not specified a default list of all mempool sample attributes as per HMC version 7.2.0 are gathered.
HMCOption-procpoolSample	A delimited separated list (comma by default) of the attribute names for the required attribute values to be gathered for the procpool sample events. If not specified a default list of all procpool sample attributes as per HMC version 7.2.0 are gathered.
HMCOption-lparSampleCmd	Switch the command on or off for lpar sample events. True is on and is the default value.
HMCOption-sysSampleCmd	Switch the command on or off for sys sample events. True is on and is the default value.
HMCOption-memPoolSampleCmd	Switch the command on or off for mempool sample events. True is on and is the default value.
HMCOption-procPoolSampleCmd	Switch the command on or off for procpool sample events. True is on and is the default value.
HMCOption-startTime	Start time of the day to gather the HMC events from. The format is: 'HH:mm'. Default is '00:00'. The Collector "logdate" parameter defines the date.
HMCOption-endTime	End time of the day to gather the HMC events to. The format is: 'HH:mm'. Default is '23:59'. The Collector "logdate" parameter defines the date.
outputFolder	This is the path to where the data file(s) from the HMC collection are written. The default path is: %CollectorFolder%/hmc. A file is generated for each managed system per HMC as follows: <OutputFolder>/<HMC_HostName>/ hmc_<logdate_starttime>_ <system_name>.txt.

The SampleHMC.xml file contains the following parameters that are specific to HMCINPUT parsing.

```
<Input name="CollectorInput" active="true">
 <Collector name="HMCINPUT">
  <Template filePath="%HomePath%/collectors/HMC/StandardTemplates.xml"
   name="HMC_LPAR_SAMPLE" />
 </Collector>
 <Parameters>
  <Parameter name="LogDate" value="PREMON" DataType="String"/>
 </Parameters>
 <Files>
  <File name="%CollectorLogs%/hmc/CurrentCSR.txt" type="input"/>
  <File name="%ProcessFolder%/LparSampleException.txt" type="exception" />
 </Files>
</Input>
```

Table 91. HMCINPUT Parameters

Parameter	Description/Values
filePath="template_file_path	The path of the template file. If you do not specify a value, the default SCCM_install_dir/collectors/HMC/StandardTemplates.xmlis used. You can generate custom template files and use them in place of the standard templates by specifying a different filename.
name="template_name"	The name of the template. This value controls the type of log records which are read and processed by the job. Valid values are HMC_LPAR_SAMPLE for LPAR Sample, HMC_MEMPOOL_SAMPLE for Mempool Sample, HMC_PROCPOOL_SAMPLE for Procpool Sample and HMC_SYS_SAMPLE for System Sample.

HMC data commands

This topic lists the commands run on the HMC. Refer to the IBM HMC documentation for more details.

The IBM HMC collector gathers usage data from the HMC using the following commands:

- lssyscfg -r sys:
  - Discover the managed systems of the HMC
- lslparutil -r config -m <managed_system>:
  - Check if data is currently being persisted by the HMC for this managed system.
- `lslparutil -r lpar –m <managed_system> <day start/end> --startyear <usage_period_year> --startmonth <usage_period_month>; --startday <usage_period_day> --starthour 0 --startminute 0 --endyear <usage_period_year> --endmonth <usage_period_month> --endday <usage_period_day+1> --endhour 00 --endminute 01 </p> -s <sampling event file> --filter "event_types=sample" –F <attributes>:`
  - LPAR data
- `lslparutil -r procpool –m <managed_system> --startyear <usage_period_year> --startmonth <usage_period_month> --startday <usage_period_day> --starthour 0 --startminute 0 --endyear <usage_period_year> --endmonth <usage_period_month>--endday <usage_period_day+1> --endhour 00 --endminute 01<day start/end> -s <sampling event file> "event_types=sample" –F <attributes>:`
  - Processor shared pool data
- `lslparutil -r mempool –m <managed_system> --startyear <usage_period_year> --startmonth <usage_period_month> --startday <usage_period_day> --starthour 0 --startminute 0 --endyear <usage_period_year> --endmonth <usage_period_month> --endday <usage_period_day+1> --endhour 00 --endminute 01</p><day start/end> -s <sampling event file> "event_types=sample" –F <attributes>:`
  - Memory shared pool data
- `lslparutil -r sys –m <managed_system> --startyear <usage_period_year> --startmonth <usage_period_month> --startday <usage_period_day> --starthour 0 --startminute 0 --endyear <usage_period_year> --endmonth`

<usage_period_month> --endday <usage_period_day+1> --endhour 00 --endminute 01 -s <sampling event file> "event_types=sample" –F <attributes>:

– Managed system data

lslparutil -r lpar -m <managed_system> -s h --filter "event_types= state_change" --endyear <usage_period_year> --endmonth <usage_period_month> --endday <usage_period_day> --endhour 00 --endminute 01:

- LPAR state changes prior to the usage period
  - lslparutil -r lpar -m <managed_system> -s h --filter "event_types= state_change" --startyear <usage_period_year> --startmonth <usage_period_month> --startday <usage_period_day> --starthour 0 --startminute 0 --endyear <usage_period_year> --endmonth <usage_period_month> --endday <usage_period_day+1> --endhour 00 --endminute 01:
- LPAR state changes during the usage period
  - lssyscfg -m <managed_system> -r lpar -F lpar_id:name:logical_serial_num:lpar_env:state:
- Current state of the LPARs

Note: The commands can be run on the HMC without the attributes specified. This will show all attributes supported for a HMC version for that event and resource type. The default attributes used by the HMC collector are specified in the sample job file. Commands and attributes for HMC events can be configured if they are not supported in a HMC version. Refer to "Setting up SmartCloud Cost Management for HMC data collection" on page 278 and IBM HMC documentation for more details.

KVM data collector

The Kernel-based Virtual Machine (KVM) collector collects virtualization information from the Linux KVM hypervisor.

The SmartCloud Cost Management KVM collector will collect information for the following KVM systems:

- RHEL KVM

The SmartCloud Cost Management KVM collector provides scripts which are scheduled in the root's crontab to generate daily files of KVM virtualization. The file contains a snapshot of the KVM virtualization usage expressed in CSR record format. The collector script provides a means to transfer the daily file to the SmartCloud Cost Management Server for processing. The KVM virtualization usage file is processed using a SmartCloud Cost Management jobrunner job file. The SmartCloud Cost Management Server provides the following files to implement the SmartCloud Cost Management KVM collector:

```
.../collectors/KVM/kvm_collect.template
.../collectors/KVM/kvm_collect.pl
.../collectors/KVM/tuam_unpack_kvm
.../collectors/KVM/KVM_DeploymentManifest_Linux.xml

.../samples/jobfiles/SampleDeployKVMCollector.xml
.../samples/jobfiles/SampleKVM.xml
```

Identifiers and resources collected by the KVM collector

Identifiers and resources collected by the Volume Manager collector are converted into a CSR file which can be consumed by SmartCloud Cost Management.

The following identifiers and resources are included in the CSR file.

Identifiers

Table 92. KVM Identifiers

Identifier	Description
HOST_NAME	Fully qualified host name
IP_ADDRESS	Host name IP address
MODEL_NAME	Host model name
SERIAL_NUM	Host manufacturer serial number
TYPE_MODEL	Host 7 character type and model
VM_NAME	Virtual Machine (VM) Name
VM_PID	The process ID of the process that runs on the hypervisor and embodies the VM
VM_SHORT_ID	The short numeric ID of the VM
VM_UUID	The universal unique ID of the VM

Resources

Table 93. KVM resources

Resource	Description
KVMHT001	Host CPU clock speed (MHz)
KVMHT002	Host number of physical CPUs
KVMHT003	Host number of CPU cores
KVMHT004	Host number of CPU threads (Depending if CPUs support hyper-threading; otherwise this is the same as KVMHT003.)
KVMHT005	Host Memory (MB)
KVMHT006	Host CPU idle time (Seconds)
KVMHT007	Host CPU other time (seconds)
KVMHT008	Host CPU system time (seconds)
KVMHT009	Host CPU user time (seconds)
KVMVM001	The sum of KVMVM003 over all VMs for the given sampling interval (This is guaranteed to be greater than zero. If all VMs have KVMVM003= 0 for a given sampling interval, each KVMVM003 in that interval is arbitrarily changed to 0.00001, and the KVMVM003 value is computed as the sum of those.)
KVMVM002	Number of virtual CPUs assigned to the VM
KVMVM003	Virtual CPU time used by the VM (Seconds)
KVMVM004	Maximum memory available to VM (MB)
KVMVM005	Memory assigned to VM (MB)
RX_BYTES	Number of bytes received (GB)

Table 93. KVM resources (continued)

Resource	Description
RX_PACKS	Number of packets received
RX_ERRS	Number of receive packet errors
RX_DROP	Number of receive packets dropped
TX_BYTES	Number of bytes transferred (GB)
TX_PACKS	Number of packets transferred
TX_ERRS	Number of transfer packet errors
TX_DROP	Number of transfer packets dropped
LOGSIZE	Log size of virtual image (KB)
DISKSZ	Size of virtual image on disk (KB)

Deploying the SmartCloud Cost Management KVM collector

The SmartCloud Cost Management KVM collector can be deployed in two ways. First, using the RXA capability built into the SmartCloud Cost Management Server. This method allows the user to install the KVM collector to the target server from the SmartCloud Cost Management Server. A second method would be to manually copy the KVM installation files to the target server and run the **tuam_unpack_kvm** installation script.

Note: The target server requires Perl to be installed and on the system path.

If the daily files are to be transferred using the standard provided means, then the collector requires that the SmartCloud Cost Management Server is running an FTP service to receive the nightly KVM usage files from the target servers.

Installing the SmartCloud Cost Management KVM collector from the SmartCloud Cost Management Server:

Installing the KVM collector from the SmartCloud Cost Management Server is accomplished using the `SampleDeployKVMCollector.xml` jobfile.

Before you begin

Several parameter settings need to be edited prior to running the jobfile, these include:

- **Host**: The hostname or IP address of the target server.
- **UserId**: Must be set to "root".
- **Password**: Password for root on the target server.
- **Manifest**: manifest file for the OS type of the target server
- **RPDParameters**: see note in the example.

Example

Example from `SampleDeployKVMCollector.xml` jobfile:

Note: Example lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
<!-- SUPPLY hostname OF TARGET PLATFORM/-->
                <Parameter Host             = "TARGET_PLATFORM"/>

                <!-- userid must be set to root/-->
```

```
                <Parameter UserId          = "root"/>

                <!-- SUPPLY root PASSWORD ON TARGET PLATFORM/-->
                <Parameter Password        = "XXXXXX"/>

                <!--Parameter KeyFilename     = "yourkeyfilename"/-->

                <!-- DEFINE Manifest TO MANIFEST XML FOR TARGET PLATFORM/-->
                <Parameter Manifest            = " KVM_DeploymentManifest_Linux.xml"/>

                <!--Parameter Protocol       = "win | ssh"/-->

                <!-- DEFINE INSTALLATION PARAMETERS,
                       path:   must be defined to the directory path where kvm Collector
                               will be installed on target platform.
                               Parameter Required.
                       server: name or IP address of the SmartCloud Cost Management Server
                               Parameter Required if do_ftp is set.
                       log_folder: CollectorLog folder on SmartCloud Cost Management Server.
If SmartCloud Cost Management Server is
              UNIX/Linux platform, log_folder should be defined to the full path to the
                               Collector Logs folder, example: /opt/ibm/tuam/logs/collectors .
                               Parameter Required if do_ftp is set.
                       ftp_user: Account used to access the SmartCloud Cost Management Server for nightly transfer of the
                               kvm log to the SmartCloud Cost Management Server.  If the log file is to remain on the
                               client, set ftp_user=HOLD. If anonymous ftp access has been configured
                               on the SmartCloud Cost Management Server, enter ftp_user=anonymous. This account must have
                               read/write access to the Collector Logs folder.
                               Parameter Required if do_ftp is set.
                       ftp_key: password used by ftp_user. If ftp_user=anonymous or ftp_user=HOLD,
                               enter any value for this parameter, but enter something,
                               example: ftp_key=XXXX.
                               Parameter Required if do_ftp is set.
                       add_ip: Indicate if the name of the folder receiving the kvm files on
                               the SmartCloud Cost Management Server should include the IP address of the client server.
                               example: If log_folder=/opt/ibm/tuam/logs/collectors and the
                               client server  is named "client1", the nightly kvm file will be
                               placed in
                                   /opt/ibm/tuam/logs/collectors/kvm/client1
                               If add_ip=Y, the file will be delivered to
                                   /opt/ibm/tuam/logs/collectors/kvm/client1_<ipaddress>.
                               Default value "N".
                       do_ftp: Perform file transfer of collector daily data using
FTP to SmartCloud Cost Management serve
                                   If do_ftp=Y, the file will be transferred.
                                   Default value "Y".
                       file_cleanup: Number of days to maintain files on client server.
             Default value "45".
                   /-->
                   <Parameter RPDParameters = "path=/opt/ibm/tuam/collectors/kvm;server=TUAM_SERVER;
log_folder=/opt/ibm/tuam/logs/collectors;ftp_user=ituam;ftp_key=XXXXXX;add_ip=N;do_ftp=Y;file_cleanup=45;"/>
```

Manually installing the SmartCloud Cost Management KVM collector:

This topic describes how to manually install the KVM collector.

About this task

The SmartCloud Cost Management KVM can be installed manually by copying the collectors installation files to the target server, placing them in the directory where you want the collector to reside and then executing the collector unpack script.

For example, suppose your SmartCloud Cost Management Server is installed in `<SCCM_install_dir>`, and you want the SmartCloud Cost Management KVM Collector to reside in `/opt/ibm/tuam/collectors/kvm` on the target server, to complete the installation you would do the following:

1. Copy `<SCCM_install_dir>/collectors/KVM/kvm_collect.template` to `/opt/ibm/tuam/collectors/kvm` on the target server.
2. Copy `<SCCM_install_dir>/collectors/KVM/kvm_collect.pl` to `/opt/ibm/tuam/collectors/kvm` on the target server.
3. Copy `<SCCM_install_dir>/collectors/KVM/tuam_unpack_kvm` to `/opt/ibm/tuam/collectors/kvm` on the target server.
4. On the target server, as root, enter the following. (Note: the values supplied are described in the installing topic. This describes how to install using the provided file transfer of FTP).

   Note: Code lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
# cd  /opt/ibm/tuam/collectors/kvm
# chmod 770 tuam_unpack_kvm
# ./tuam_unpack_kvm  path=/opt/ibm/tuam/collectors/kvm  server=tuamserver \
log_folder=collector_log  ftp_user=ituam  ftp_key=ituam  add_ip=N do_ftp=Y \
file_cleanup=45
```

Following the installation of the SmartCloud Cost Management KVM collector:

This topic discusses the actions that must be taken after you have installed the KVM collector.

The installation places an entry in the root crontab to run the `kvm_collect.sh` script once each hour. The script will output hourly KVM virtualization information into the host and guest data: `SCCM_install_dir/kvm_host_YYYYMMDD_HH.csv` and `SCCM_install_dir/kvm_guest_YYYYMMDD_HH.csv`.

When called at 12 midnight, the script will concatenate the hourly virtualization information for the previous day into single files for guest and host `kvm_host_YYYYMMDD.csv` and `kvm_host_YYYYMMDD.csv`. If enabled, this file is then transferred to the SmartCloud Cost Management server and placed in the path configured during deployment. for example, `<collector_logs>/kvm/<target_name>`. A log of the collection is written to `/<SCCM_install_dir>/log/kvm_collector_YYYYMMDD.log`. A log of the FTP transfer is written to `/<SCCM_install_dir>/data/YYYYMMDD_ftp.log`.

The daily KVM usage files and ftp logs are retained on the target platform for 45 days.

Script - kvm_collect.sh

Usage:

```
kvm_collect.sh  collect  |  send [YYYYMMDD]
```

The `kvm_collect.sh` script is called with either the collect or sends argument.

collect: This argument should only be used in the crontab entry that is placed in `roots` cron file when the KVM is installed. A log of the collection is written to `/<SCCM_install_dir>/log/kvm_collector_YYYYMMDD.log`.

send: This argument can be used to send the nightly KVM report file to the SmartCloud Cost Management Server. A log of the FTP transfer is written to `/<SCCM_install_dir>/data/YYYYMMDD_ftp.log`.

Uninstalling the SmartCloud Cost Management KVM collector:

This topic describes how to uninstall the KVM collector.

About this task

The SmartCloud Cost Management KVM collector can be uninstalled as follows:

Procedure

1. Remove the `kvm_collect.sh crontab` entry from root's crontab.
2. Remove all KVM files and scripts from the path the collector was installed, for example, `/opt/ibm/collectors/kvm`.

Linux, z/Linux, UNIX, and AIX operating system and file system data collectors

If you installed the operating system and file system data collectors remotely, data collection is started automatically.

See "Setting up operating and file system data collection: starting Linux or UNIX process accounting" on page 394.

Sar data collector

Sar is a command available on all UNIX/Linux platforms which provides the user with system-level CPU utilization and capacity information.

Sar reports information in time intervals as requested by the user. For example, the user may request sar to report 10 5-minute intervals of utilization and capacity information.

```
# sar 300 10
```

The SmartCloud Cost Management sar collector provides a script which is scheduled in root's crontab to generate sar reports and transfer the reports to the SmartCloud Cost Management Server on a daily basis for processing. The sar reports are processed using a SmartCloud Cost Management jobrunner job file. SmartCloud Cost Management Server provides the following files to implement the SmartCloud Cost Management sar Collector.

```
.../collectors/sar/sar_collect.template
.../collectors/sar/tuam_unpack_sar
.../collectors/sar/sarDeploymentManifest_linux.xml
.../collectors/sar/sarDeploymentManifest_hp.xml
.../collectors/sar/sarDeploymentManifest_aix.xml
.../collectors/sar/sarDeploymentManifest_solaris.xml

.../samples/jobfiles/SampleDeploySarCollector.xml
.../samples/jobfiles/SampleSar.xml
```

SmartCloud Cost Management products

The following is a list of the SmartCloud Cost Management products that include this collector:

SmartCloud Cost Management products
• IBM SmartCloud Cost Management

Identifiers and resources collected by sar collector

Identifiers and resources collected by the sar collector are converted to CSR record format by an Integrator stage in the SampleSar.xml jobfile.

The following identifiers and resources are included in the CSR file.

DITA

Identifiers:

Table 94. Sar Identifiers

Identifier	Description
SYSTEM_ID	Servername

Table 94. Sar Identifiers (continued)

Identifier	Description
IP_ADDR	IP address of server
OP_SYS	Operating system of server
INT_START	Interval start time YYYYMMDDHHMMSS
INT_END	Interval end time YYYYMMDDHHMMSS

Resources:

Table 95. Sar Resources

Resource	Description
SANUMCPU	Number physical CPUs
SAMEMSIZ	Memory Size (KB)
SAUCPUPT	Percent User CPU
SASCPUPT	Percent System CPU
SAICPUPT	Percent Idle CPU
SAWAIOPT	Percent Wait I/O
SAUCPUUS	User CPU used (seconds)
SASCPUUS	System CPU used (seconds)
SAICPUUS	Idle CPU Time (seconds)
SAWAIOUS	Wait I/O Time (seconds)
SAENTCPT	Percent Entitled Capacity

It is recommended that this data is written to the resource utilization table. For more information on how to do this, see the DBLoad section.

Deploying the sar collector

The SmartCloud Cost Management sar Collector can be deployed in one of two ways. First, using the RXA capability built into SmartCloud Cost Management Server. This method allows the user to install the sar collector to the target server from the SmartCloud Cost Management Server. A second method would be to manually copy the sar installation files to the target server and run the tuam_unpack_sar installation script.

Regardless of the method used to install, the collector requires that the SmartCloud Cost Management Server is running an FTP service to receive the nightly sar usage files from the target servers. Where SmartCloud Cost Management server is running on Windows, you will need to be sure of the following:

- The IIS FTP service is installed.
- A virtual directory has been defined to the SmartCloud Cost Management Collector Logs folder.
- The FTP service can be configured to allow either anonymous or user/password access, but the accessing account must have read/write access to the SmartCloud Cost Management Collector Logs folder.

Installing the SmartCloud Cost Management sar Collector from the SmartCloud Cost Management Server:

Installing the sar Collector from the SmartCloud Cost Management Server is accomplished using the `SampleDeploySarCollector.xml` jobfile.

Before you begin

Several parameter settings need to be edited prior to running the jobfile.

- **Host**: The hostname or IP address of the target server.
- **UserId**: Must be set to "root".
- **Password**: Password for root on the target server.
- **Manifest**: manifest file for the OS type of the target server
- **RPDParameters**: see note in the example.

Example

Example from `SampleDeploySarCollector.xml` jobfile:

Note: Example lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
                <!-- SUPPLY hostname OF TARGET PLATFORM/-->
                <Parameter Host             = "TARGET_PLATFORM"/>

                <!-- userid must be set to root/-->
                <Parameter UserId           = "root"/>

                <!-- SUPPLY root PASSWORD ON TARGET PLATFORM/-->
                <Parameter Password         = "XXXXXX"/>

                <!--Parameter KeyFilename     = "yourkeyfilename"/-->

                <!-- DEFINE Manifest TO MANIFEST XML FOR TARGET PLATFORM/-->
                <!--Parameter Manifest         = "sarDeploymentManifest_linux.xml"/-->
                <!--Parameter Manifest         = "sarDeploymentManifest_hp.xml"/-->
                <!--Parameter Manifest         = "sarDeploymentManifest_linux.xml"/-->
                <!--Parameter Manifest         = "sarDeploymentManifest_solaris.xml"/-->
                <Parameter Manifest         = "sarDeploymentManifest_aix.xml"/>

                <!--Parameter Protocol       = "win | ssh"/-->

                <!-- DEFINE INSTALLATION PARAMETERS,
                       path:   must be defined to the directory path where sar Collector
                               will be installed on target platform.
                               Parameter Required.
                       server: name or IP address of the
SmartCloud Cost Management Server
                               Parameter Required.
                       log_folder: CollectorLog folder on SmartCloud Cost Management Server.
If SmartCloud Cost Management Server is UNIX/Linux
                               platform, log_folder should be defined to the full path to the
                               Collector Logs folder, example: /opt/ibm/tuam/logs/collectors .
                               If SmartCloud Cost Management Server is Windows platform, log_folder should be set to
                               virtual directory that references the Collector Logs folder.
                               Parameter Required.
                       ftp_user: Account used to access the SmartCloud Cost Management Server for nightly transfer of the
                               sar log to the
SmartCloud Cost Management Server.  If the log file is to remain on the
                               client, set ftp_user=HOLD. If anonymous ftp access has been configured
                               on the SmartCloud Cost Management Server, enter ftp_user=anonymous. This account must have
                               read/write access to the Collector Logs folder.
                               Parameter Required.
                       ftp_key: password used by ftp_user. If ftp_user=anonymous or ftp_user=HOLD,
                               enter any value for this parameter, but enter something,
                               example: ftp_key=XXXX.
                               Parameter Required.
                       add_ip: Indicate if the name of the folder receiving the sar files on
                               the TUAM Server should include the IP address of the client server.
                               example: If log_folder=/opt/ibm/tuam/logs/collectors and the
                               client server  is named "client1", the nightly sar file will be
                               placed in
                                   /opt/ibm/tuam/logs/collectors/sar/client1
                               If add_ip=Y, the file will be delivered to
                                   /opt/ibm/tuam/logs/collectors/sar/client1_<ipaddress>.
                               Default value "N".
                       interval: Number of seconds in sar intervals.
                               Default value 300.
                -->
            <Parameter RPDParameters    = "path=/data/tuam/collectors/sar;server=9.42.17.133; \
    log_folder=/data/tuam/logs/collectors;ftp_user=ituam;ftp_key=ituam;add_ip=Y;interval=300;"/>
```

Manually installing the SmartCloud Cost Management sar collector:

This topic describes how you can manually install the sar collector.

About this task

The SmartCloud Cost Management sar Collector can be installed manually by copying the collectors installation files to the target server, placing them in the directory where you want the collector to reside and then executing the collector unpack script.

For example, suppose your SmartCloud Cost Management Server is installed in <SCCM_install_dir>, and you want the SmartCloud Cost Management sar collector to reside in /opt/ibm/tuam/collectors/sar on the target server, to complete the installation you would do the following:

1. Copy <SCCM_install_dir>/collectors/sar/sar_collect.template to /opt/ibm/tuam/collectors/sar on the target server.
2. Copy <SCCM_install_dir>/collectors/sar/tuam_unpack_sar to /opt/ibm/tuam/collectors/sar on the target server.
3. On the target server, as root, enter the following. (Note: the values supplied are described in the installing topic).

   Note: Code lines that are too long are broken up. The breakpoint is indicated using a backslash "/".

```
# cd/opt/ibm/tuam/collectors/sar
# chmod 770 tuam_unpack_sar
# ./tuam_unpack_sar  path=/opt/ibm/tuam/collectors/sar
server=tuamserver log_folder=collector_log  ftp_user=ituam  ftp_key=ituam  add_ip=N  interval=300
```

Following the installation of the sar collector:

This topic discusses the actions that must be taken after you have installed the sar collector.

The installation places an entry in the root crontab to run the sar_collect.sh script once each hour. The script will output an hourly sar report to `/<install_dir>/data/sar_YYYYMMDD_HH.txt`.

When called in the 23rd hour (11 PM), the script will generate the hourly report and then concatenate the hourly sar reports into a single file `sar_YYYYMMDD.txt`. This file is then transferred to the SmartCloud Cost Management Server and placed in `<collector_logs>/sar/<target_name>`. A log of the FTP transfer is written to `/<install_dir>/data/YYYYMMDD_ftp.log`

The daily sar usage files and ftp logs are retained on the target platform for 10 days.

Script - sar_collect.sh

Usage:

```
sar_collect.sh  collect  |  send [YYYYMMDD ]
```

The `sar_collect.sh` script is called with either the collect or send argument.

collect : This argument should only be used in the crontab entry that is placed in root's cron file when the sar collector is installed.

send : This argument can be used to send the nightly sar report file to the SmartCloud Cost Management Server. A log of the FTP transfer is written to `/<install_dir>/data/YYYYMMDD_ftp.log`.

Uninstalling the sar collector:

This topic describes how to uninstall the sar collector.

About this task

The SmartCloud Cost Management sar Collector can be uninstalled simply by removing the `sar_collect.sh` crontab entry from root's crontab. Once that is done, all sar collector files and scripts can be removed.

Tivoli Data Warehouse data collector

You can use the Tivoli Data Warehouse data collector to gather usage and accounting data that is stored in the Tivoli Data Warehouse.

IBM Tivoli Monitoring manages IT infrastructure, including operating systems, databases, and servers, across distributed and host environments through a single workspace portal. Tivoli Monitoring agents collect data and store it in Tivoli Data Warehouse. The Tivoli Data Warehouse collector collects usage and accounting data that is gathered by IBM Tivoli Monitoring agents and stored in the Tivoli Data Warehouse and converts the data to Common Source Record format.

Setting up Tivoli Data Warehouse data collection

This topic provides information about setting up Tivoli Data Warehouse data collection.

About this task

SmartCloud Cost Management provides several sample job files that you can modify and use to collect data from Tivoli Data Warehouse. After you have modified and optionally renamed a sample job file, move the file to the `<SCCM_install_dir>\jobfiles` directory.

Configuring the Tivoli Data Warehouse job file

SmartCloud Cost Management provides several sample job files for Tivoli Data Warehouse data collection. These sample files provide a template that you can use to create a job file.

The following table lists the sample job files for Tivoli Data Warehouse data collection.

Table 96.

File name	Description
SampleTDW_AEM.xml	Active Energy Manager
SampleTDW_AIXPremium.xml	AIX Premium
SampleTDW_Eaton.xml	Eaton (Power)
SampleTDW_HMC.xml	Hardware Management Console for p5, i5 or OpenPower™ systems
SampleTDW_LinuxOS.xml	Linux Operating System
SampleTDW_TCAM.xml	IBM Tivoli Composite Application Manager for SOA

Table 96. (continued)

File name	Description
SampleTDW_UnixOS.xml	Unix Operating System
SampleTDW_WindowsOS.xml	Windows Operating System

The comments that are included in the sample job file contain additional information about how to customize the file for your environment. It is recommended that you copy the sample job file to the `<SCCM_install_dir>\jobfiles` directory where you can make any necessary modifications before scheduling the job to run.

Structure of the job file

This section describes the job file for the Tivoli Data Warehouse collector.

The following sample shows the structure of the Integrator step of the job file:

```
<Step id="Integrator-ITMPWMRS" type="ConvertToCSR" programType="java" programName="integrator"  active="true">
              <Integrator>
                 <Input name="CollectorInput" active="true">

                    <Collector name ="TDW">
                       <Connection dataSourceName="OCEAN5"/>
                       <Template filePath="%HomePath%/collectors/tdw/StandardTemplates.xml" name="TDWAEMRS" />
                    </Collector>

                    <Parameters>
                       <Parameter name="StartLogDate" value="%LogDate_Start% 00:00:00" dataType="DATETIME" format="yyyyMMdd HH:mm:ss"/>
                       <Parameter name="EndLogDate" value="%LogDate_End% 23:59:59" dataType="DATETIME" format="yyyyMMdd HH:mm:ss"/>
                       <!--Parameter name="Resourceheader" value="TDWAAEMRS" dataType="STRING"/-->
                       <Parameter name="Feed" value="server1" dataType="STRING"/>
                       <Parameter name="LogDate" value="%LogDate_End%" dataType="DATETIME" format="yyyyMMdd"/>
                    </Parameters>

                    <Files>
                       <File name="%ProcessFolder%/exception.txt" type="exception" />
                    </Files>
```

The Step element has the following attributes:

Table 97.

Attribute	Required or optional	Description/Parameters
id="*step_id*"	Required.	*step_id* refers to the step ID.
type="ConvertToCSR"	Required.	Do not change this value.
programType="java"	Required.	Do not change this value.
programName="integrator"	Required.	Do not change this value.

The Step element contains the Integrator element, which has no attributes.

The Integrator element contains the Input element, which has the following attributes:

Table 98.

Attribute	Required or optional	Description/Parameters
name="CollectorInput"	Required.	Do not change this value.
active="true" | "false"	Required.	Set this attribute to "true" to run the collector or "false" if you do not want to run the collector.

The Input element can contain the Collector, Parameters, and Files elements.

The Collector element is required. It has one attribute:

Table 99.

Attribute	Required or optional	Description/Parameters
name="TDW"	Required.	Do not change this value.

The Collector element contains the Connection and Template elements.

The Connection element is required and has the following attribute:

Table 100.

Attribute	Required or optional	Description/Parameters
dataSourceName="*data_source_name*"	Required.	The name of the data source defined in the Data Source List Maintenance page in the Administration Console.

The Template element is required and has the following attributes:

Table 101.

Attribute	Required or optional	Description/Parameters
filePath="*template_file_path*"	Optional.	The path of the template file. If you do not specify a value, the default `%HomePath%/collectors/tdw/StandardTemplates.xml` is used. You can generate custom template files and use them in place of the standard templates by specifying a different file name.
name="*template_name*"	Required.	The name of the template.

The Parameters element has no attributes. The Parameters element must contain the startLogDate, endLogDate, Feed, and LogDate parameters and may contain the optional parameter Resourceheader.

The startLogDate parameter contains the following attributes:

Table 102.

Attribute	Required or optional	Description/Parameters
name="startLogDate"	Required.	The name of the parameter.
value="*interval_start_date_time*"	Required.	The start time of the usage interval.
dataType="DATETIME"	Required.	The data type expected for this parameter.
format="yyyyMMdd HH:mm:ss"	Required.	The simple date time format that is used in the date time value.

The endLogDate parameter contains the following attributes:

Table 103.

Attribute	Required or optional	Description/Parameters
name="endLogDate"	Required.	The name of the parameter.
value="*interval_end_date_time*"	Required.	The end time of the usage interval.

Table 103. (continued)

Attribute	Required or optional	Description/Parameters
dataType="DATETIME"	Required.	The data type expected for this parameter.
format="yyyyMMdd HH:mm:ss"	Required.	The simple date time format that is used in the date time value.

The optional Resourceheader parameter contains the following attributes:

Table 104.

Attribute	Required or optional	Description/Parameters
name="Resourceheader"	Required.	The name of the parameter.
value="*header_value*"	Required.	The resource header. This attribute specifies the value to be placed in the header field of the CSR output file. The default value is the first eight characters of the name attribute specified in the Template element.
dataType="STRING"	Required.	The data type.

The Feed parameter contains the following attributes:

Table 105.

Attribute	Required or optional	Description/Parameters
name="Feed"	Required.	The name of the parameter.
value="server1"	Required.	Specifies the subdirectory of the process folder where CSR or CSR+ files from the same feed are stored.
dataType="STRING"	Required.	The data type.

The LogDate parameter contains the following attributes:

Table 106.

Attribute	Required or optional	Description/Parameters
name="LogDate"	Required.	The name of the parameter.
value="%LogDate_End%"	Required.	The date for which you are processing usage logs. The value can be a date literal or a date keyword.
dataType="DATETIME"	Required.	The data type expected for this parameter.
format="yyyyMMdd"	Required.	The simple date time format that is used in the date time value.

The Files element is optional. It has no attributes. The Files element may contain the File parameter. The File parameter has the following attributes:

Table 107.

Attribute	Required or optional	Description/Parameters
name="*exception_file_path*"	Required.	The name of a file in which to log exceptions.
type="exception"	Required.	The type of data written to the file.

Active Energy Manager identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Active Energy Manager agent.

Table 108. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Active Energy Manager agent

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
Active Energy Manager	Rack Server	KE9_ALL_RACK_SERVERS_POWER_DATA Resources • Average_Power_AC (INTEGER, average) • Average_Power_DC (INTEGER, average) • Ambient_Temp_Avg (DECIMAL, average) • Exhaust_Temp_Avg (DECIMAL, average) IDs • Name • Type • OID	 Resources PWRAC (KWH) PWRDC (KWH) PWRAMTMP PWREXTMP IDs name server_type server_oid agent = “aem” attribute_group = “rack_server”

Table 108. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Active Energy Manager agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Blade Slot	KE9_ALL_BLADECENTERS_POWER_DATA	
		Resources • Average_Power_DC	Resources PWRDC (KWH)
		(INTEGER, average) • Ambient_Temp_Avg	
		(DECIMAL, average) • Exhaust_Temp_Avg	PWRAMTMP
		(DECIMAL, average)	PWREXTMP
		IDs • Slot_Number • Name • Domain • Component_ID • Component_Type • OID	IDs slot_number blade_center_name power_domain_name name slot_type server_oid
			agent = “aem” attribute_group = “blade_slot”

Table 108. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Active Energy Manager agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	System Z	KE9_ALL_SYSTEM_Z10_ECS_POWER_DATA Resources • Average_Input_Power_AC (INTEGER, average) • Ambient_Temp_Avg (DECIMAL, average) • Exhaust_Temp_Avg (DECIMAL, average) IDs • Name • Type • Model • Serial_Number • OID	 Resources PWRAC (KWH) PWRAMTMP PWREXTMP IDs name mech_type model serial_num server_oid agent = “aem” attribute_group = “systemz”

Table 108. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Active Energy Manager agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	PDU Outlet Group	KE9_ALL_PDUS_POWER_DATA Resources • Input_Power (INTEGER, average) • Output_Power (INTEGER, average) • Ambient_Temp_Avg (DECIMAL, average) • Exhaust_Temp_Avg (DECIMAL, average) IDs • Name • Group_Name • URL	Resources PWRIN (KWH) PWROUT (KWH) PWRAMTMP PWREXTMP IDs pdu_name outlet_group_name url agent = “aem” attribute_group = “pdu_outlet”

AIX Premium identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent.

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
AIX Premium	File Systems	KPX_FILE_SYSTEMS Resources • Size_MB (INTEGER, snapshot) • Free_MB (INTEGER, snapshot) • Used_MB (INTEGER, snapshot) IDs • Name • Node • Mount_Point • Volume_Group_Name	 Resources AIXPFSTS (MB-Interval) AIXPFSFS (MB-Interval) AIXPFSUS (MB-Interval) IDs name node mount_point volume_group_name agent = “aix_premium” attribute_group = “file_system”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Logical Partition	KPX_LOGICAL_PARTITION Resources • Entitlement (INTEGER, snapshot) • Number_of_Physical_CPUs (INTEGER, snapshot) • Number_of_Virtual_CPUs (INTEGER, snapshot) • Number_of_Logical_CPUs (INTEGER, snapshot) • Online_Mem (INTEGER, snapshot) IDs • Node • LPAR_Number • Shared_Mode • Capped_Mode • SMT_Mode • Machine_ID • Hostname	Resources AIXPLPET AIXPLPPC AIXPLPVC AIXPLPLC AIXPLPOM (MB-Interval) IDs node lpar_number shared_mode capped_mode smt_mode machine_id hostname agent = “aix_premium” attribute_group = “logical_partition”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Logical Volume	KPX_LOGICAL_VOLUMES Resources • Size_MB (INTEGER, snapshot) IDs • Node • Name • Volume_Group_Name • Type • Mount_Point	Resources AIXPLVTS (MB-Interval) IDs node name volume_group_name type mount_point agent = “aix_premium” attribute_group = “logical_volume”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Network Adapters Totals	KPX_NETWORK_ADAPTERS_TOTALS	
		Resources • Bytes_Sent (VARCHAR, accumulation) • Pkts_Sent (VARCHAR, accumulation) • Broadcast_Pkts_Sent (VARCHAR, accumulation) • Multicast_Pkts_Sent (VARCHAR, accumulation) • Bytes_Recvd (VARCHAR, accumulation) • Pkts_Recvd (VARCHAR, accumulation) • Broadcast_Pkts_ Recvd (VARCHAR, accumulation) • Multicast_Pkts_ Recvd (VARCHAR, accumulation)	Resources AIXPNABS AIXPNAPS AIXPNARS AIXPNAMS AIXPNABR AIXPNAPR AIXPNARR AIXPNAMR
		IDs • Node • Name • Type	IDs node name AdapterType
			agent = “aix_premium” attribute_group = “network_adapters_total”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Physical Memory	KPX_PHYSICAL_MEMORY Resources • Memory_Size_MB (INTEGER, snapshot) • Free_Memory_MB (INTEGER, snapshot) • Used_Memory_MB (INTEGER, snapshot) IDs • Node	Resources AIXPPMMS (MB-Interval) AIXPPMFM (MB-Interval) AIXPPMUM (MB-Interval) IDs node agent = “aix_premium” attribute_group = “physical_memory”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Physical Volumes	KPX_PHYSICAL_VOLUMES	
		Resources	Resources
		• Number_of_Logical_Volumes (INTEGER, snapshot)	AIXPPVLV
		• Number_of_Stale_Partitions (INTEGER, snapshot)	AIXPPVSP
		• Size_MB (INTEGER, snapshot)	AIXPPVTS (MB-Interval)
		• Used_MB (INTEGER, snapshot)	AIXPPVUS (MB-Interval)
		• Free_MB (INTEGER, snapshot)	AIXPPVFS (MB-Interval)
		IDs	
		• Node	IDs
		• Name	
		• State	node
		• Volume_Group_Name	name state volume_group_name
			agent = “aix_premium”
			attribute_group = “physical_volume”

Table 109. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the AIX Premium agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Processes Detail	KPX_PROCESSES_DETAIL Resources • Text_Size (INTEGER, snapshot) • Resident_Text_Size (INTEGER, snapshot) • Resident_Data_Size (INTEGER, snapshot) • Total_CPU_Time (INTEGER, total) IDs • Node • Process_Name • User_Name	Resources AIXPPDTT (4K Page – Day) AIXPPDRT (4K Page – Day) AIXPPDRD (4K Page – Day) AIXPPDTC (Seconds) IDs node process_name user_name agent = “aix_premium” attribute_group = “process_detail”

Eaton identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Eaton agent.

Table 110. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Eaton agent

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
Eaton	PDU Breaker Meters	KE8_WH_PDU_BREAKER_METERS_TABLE Resources • TotalKilowattHours (INTEGER, accumulation) • Node • Manufacturer • Model • SerialNumber • Panel (Integer) • Breaker (Integer)	Resources ETNPBMTE(KWH) IDs node manufacturer model serial_number panel (String) breaker (String) agent = “eaton” attribute_group = “pdu_breaker_meters”
	UPS Input Table	KE8_WH_UPS_INPUT_TABLE Resources • inputWatts (INTEGER, snapshot) IDs • Node • Manufacturer • Model • SerialNumber	Resources ETNUPSIE(KWH) IDs node manufacturer model serial_number agent = “eaton” attribute_group = “ups_input_table”

Table 110. SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Eaton agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	UPS Output Table	KE8_WH_UPS_OUTPUT_TABLE Resources • outputWatts (INTEGER, snapshot) IDs • Node • Manufacturer • Model • SerialNumber	Resources ETNUPSOE(KWH) IDs node manufacturer model serial_number agent = “eaton” attribute_group = “ups_output_table”

HMC identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the HMC agent.

Table 111. Identifiers and resources that are collected from Tivoli Data Warehouse by the HMC agent.

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
HMC	Physical Memory	KPH_PHYSICAL_MEMORY	
		Resources • Total_Size_MB (INTEGER, snapshot, No data) • Used_MB (INTEGER, snapshot, No data) • Free_MB (INTEGER, snapshot, No data)	Resources HMCPMTSM HMSPMUSM HMCPMFSM
		IDs • Node	IDs node agent = “hmc” attribute_group = “physical_memory”

ITCAM/SOA identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the ITCAM/SOA agent.

Table 112. Identifiers and resources that are collected from Tivoli Data Warehouse by the ITCAM/SOA agent.

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
ITCAM/ SOA	Services Inventory Requester Identity_610	Services_Inventory_ReqID_610 Resources • Msg_Count (INTEGER, accumulation) • Elapsed_Time_Msg_Count (INTEGER, accumulation) • Fault_Count (INTEGER, accumulation) • Avg_Msg_Length (INTEGER, weighted avg) • Avg_Elapsed_Time (INTEGER, weighted avg) IDs • Origin_Node • Requester_Identity • Service_Type • Service_Name • Service_Port_Name • Operation_Name • Local_Host_Name • Local_IPAddress • Application_Server_Env • Application_Server_Node_Name • Application_Server_Cell_Name • Application_Server_Name • Port_Namespace • Port_Number • Operation_Namespace • Interval_Status	Resources TCAMMC TCAMETMC TCAMFLTC TCAMMLA (bytes) TCAMELTA (milliseconds) IDs origin_node requester_identity service_type service_name service_port_name operation_name local_host_name local_ipaddress application_server_env application_server_node_name application_server_cell_name application_server_name port_namespace port_number operation_namespace interval_status agent = “itcam_soa” `attribute_group = "services_inventory_ reqid_610"`

Linux OS identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Linux OS agent.

Table 113. Identifiers and resources that are collected from Tivoli Data Warehouse by the Linux OS agent.

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
Linux OS	Disk	Linux_Disk	
		Resources • Total_Inodes (INTEGER, snapshot) • Inodes_Used (INTEGER, snapshot) • Inodes_Free (INTEGER, snapshot) • Size (INTEGER, MB, snapshot) • Space_Used (INTEGER, MB, snapshot) • Space_Available (INTEGER, MB, snapshot)	Resources LNXDKTIN LNXDKUIN LNXDKFIN LNXDKTSS LNXDKUSS LNXDKFSS
		IDs • System_Name • Mount_Point • Disk_Name • FS_Type	IDs system_name mount_point disk_name fs_type
			agent = “linux_os” attribute_group = “disk”

Table 113. Identifiers and resources that are collected from Tivoli Data Warehouse by the Linux OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Disk IO	Linux_Disk_IO Resources • Blks_read (INTEGER, accumulation) • Blks_wrtn (INTEGER, accumulation) IDs • Dev_Major • Dev_Minor • Dev_Name • System_Name	Resources LNXDIBKR LNXDIBKW IDs device_major device_minor device_name system_name agent = “linux_os” attribute_group = “disk_io”

Table 113. Identifiers and resources that are collected from Tivoli Data Warehouse by the Linux OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Network	Linux_Network Resources • Kbytes_Received_Count (INTEGER, total) • Kbytes_Transmitted_Count (INTEGER, total) IDs • System_Name • Network_Interface_Name • Interface_IP_Address • Linux_VM_ID	Resources LNXNWRCM (MB-Interval) LNXNWTCM (MB-Interval) IDs system_name interface_name interface_ip_address vm_id agent = “linux_os” attribute_group = “network”

UNIX OS identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the UNIX OS agent.

Table 114. Identifiers and resources that are collected from Tivoli Data Warehouse by the UNIX OS agent

Tivoli Monitoring agent name	**Tivoli Monitoring attribute group**	**Database table and column names**	**Resource description and identifier name in SmartCloud Cost Management**
UNIX OS	Disk	Disk Resources • Inodes_Used (INTEGER, snapshot) • Space_Used_MB (INTEGER, MB, snapshot) IDs • System_Name • Mount_Point • Name • FS_Type • Name_U • Mount_Point_U	Resources UNXDKIU UNXDKUS IDs system_name mount_point_id name fs_type name_unicode mount_point_unicode agent = “unix_os” attribute_group = “disk”

Table 114. Identifiers and resources that are collected from Tivoli Data Warehouse by the UNIX OS agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Network	Network Resources • Received_Count (INTEGER, total) • Transmitted_Count (INTEGER, total) IDs • Network_Interface_Name • IPV4_DNS_Name • Interface_DNS_Name • Interface_IP_Address • Interface_Status • System_Name • Type	Resources UNXNWRP UNXNWTP IDs network_interface_name ipv4_dns_name interface_dns_name interface_ip_address interface_status system_name type agent = “unix_os” attribute_group = “network”

Table 114. Identifiers and resources that are collected from Tivoli Data Warehouse by the UNIX OS agent (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Solaris Zones	Solaris_Zones Resources • Physical_Memory (INTEGER, snapshot) • Total_CPUs (INTEGER, snapshot) • Virtual_Memory (INTEGER, snapshot) • Zone_CPU_Usage (Decimal, snapshot, non-negative) IDs • Init_PID • Name • Path • Pool_ID • Scheduler • Status • Zone_ID • System_Name	Resources UNXSZPM UNXSZTC UNXSZVM UNXSZCU IDs init_PID name path pool_id scheduler status zone_id system_name agent = “unix_os” attribute_group = “solaris_zones”

Windows OS identifiers and resources

This topic describes the SmartCloud Cost Management identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent.

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent.

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
Windows OS	FTP Service	FTP_Service Resources • Total_Anonymous_Users (INTEGER, accumulation) • Total_Connection_Attempts (INTEGER, accumulation) • Total_Files_Received (INTEGER, accumulation) • Total_Files_Sent (INTEGER, accumulation) • Total_Files_Transferred (INTEGER, accumulation) • Total_Logon_Attempts (INTEGER, accumulation) • Total_NonAnonymous_Users (INTEGER, accumulation) IDs • FTP_Site • System_Name	Resources WINFTPAN WINFTPCA WINFTPFR WINFTPFS WINFTPFT WINFTPLA WINFTPNU IDs ftp_site system_name agent = “windows_os” attribute_group = “ftp_server”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Job Object	Job_Object Resources • Process_Count_Total (INTEGER, accumulation) • Total_mSec_Kernal_Mode (INTEGER, msec, accumulation) • Total_mSec_Processor (INTEGER, msec, accumulation) • Total_mSec_User_Mode (INTEGER, msec, accumulation) IDs • Name • Name_U • System_Name	Resources WINJOPC WINJOKM WINJOPR WINJOUM IDs name name_unicode system_name agent = “windows_os” attribute_group = “job_object”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Logical Disk	NT_Logical_Disk Resources UsedMB (INTEGER, MB, snapshot, calculated by using Total_Size * %_Used) IDs • Disk_Name • Physical_Disk_Number • Server_Name	Resources WINLDUS IDs disk_name physical_disk_number server_name agent = “windows_os” attribute_group = “nt_logical_disk”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Print Job	NT_Print_Job (?) : Not in DB	
		Resources • Pages_Printed (INTEGER, total) • Size (INTEGER, total) • Time_Elapsed (INTEGER, second, total) • Total_Pages (INTEGER, total)	Resources WINPJPP WINPJSZ WINPJTE WINPJTP
		IDs • Data_Type • Document_Name • Document_Name_U • Driver_Name • Machine_Name • Notify_Name • Notify_Name_U • Print_Processor • Printer_Name • Printer_Name_U • Priority • System_Name • User_Name • User_Name_U	IDs data_type document_name document_name_u driver_name machine_name notify_name notify_name_u print_processor printer_name printer_name_unicode priority system_name user_name user_name_unicode
			agent = “windows_os” attribute_group = “print_job”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Print Queue	Print_Queue Resources • Total_Jobs_Printed (INTEGER, accumulation, No data) • Total_Pages_Printed (INTEGER, accumulation, No data) IDs • Name • Name_U • System_Name	Resources WINPQJP WINPQPP IDs name name_unicode system_name agent = “windows_os” attribute_group = “print_queue”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Process	NT_Process Resources • Elapsed_Time (INTEGER, snapshot) • Page_File_kBytes (INTEGER, KB, snapshot) • Page_File_kBytes_Peak (INTEGER, KB, snapshot) • Pool_Nonpaged_Bytes (INTEGER, Byte, snapshot) • Pool_Paged_Bytes (INTEGER, Byte, snapshot) • Private_kBytes (INTEGER, Byte, snapshot) • Virtual_kBytes (INTEGER, KB, snapshot) • Virtual_kBytes_Peak (INTEGER, KB, snapshot)	Resources WINPCET WINPCPF WINPCPFP WINPCPN WINPCPP WINPCPV WINPCVT WINPCVTP
	Process (cont.)	IDs • Binary_Path • ID_Process • Priority_Base • Process_Name • Server_Name • User	IDs binary_path process_id priority_base process_name server_name user agent = “windows_os” attribute_group = “nt_process”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	SMTP Server	SMTP_Server	
		Resources • kBytes_Received_Total (INTEGER, accumulation, No data) • kBytes_Sent_Total (INTEGER, accumulation, No data) • kBytes_Total (INTEGER, accumulation, No data) • Directory_Drops_Total (INTEGER, accumulation, No data) • ETRN_Messages_Total (INTEGER, accumulation, No data) • Inbound_Connections_Total (INTEGER, accumulation, No data) • Message_kBytes_Received_Total (INTEGER, accumulation, No data) • Message_kBytes_Sent_Total (INTEGER, accumulation, No data) • Message_kBytes_Total (INTEGER, accumulation, No data) • Message_Delivery_Retries (INTEGER, accumulation, No data) • Message_Send_Retries (INTEGER, accumulation, No data) • Messages_Delivered_Total (INTEGER, accumulation, No data)	Resources WINSSRS WINSSSS WINSSSZ WINSSDD WINSSEM WINSSIC WINSSMRS WINSSMSS WINSSMS WINSSMDR WINSSMSR WINSSMD

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	SMTP Server (cont.)	• Messages_Received_Total (INTEGER, accumulation, No data) • Messages_Refused_for_Size (INTEGER, accumulation, No data) • Messages_Retrieved_Total (INTEGER, accumulation, No data) • Messages_Sent_Total (INTEGER, accumulation, No data) • NDRs_Generated (INTEGER, accumulation, No data) • Outbound_Connections_Refused (INTEGER, accumulation, No data) • Outbound_Connections_Total (INTEGER, accumulation, No data) • Routing_Table_Loopups_Total (INTEGER, accumulation, No data) • Total_Connection_Errors (INTEGER, accumulation, No data) IDs • SMTP_Server • System_Name	WINSSMR WINSSMRF WINSSMRT WINSSMSN WINSSNG WINSSOCR WINSSOCN WINSSRTL WINSSCEN IDs smtp_name system_name agent = “windows_os” attribute_group = “smtp_server”

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Web Service	Web_Service Resources • Total_CGI_Requests (INTEGER, accumulation, No data) • Total_Connection_Attempts (INTEGER, accumulation, No data) • Total_Files_Received (INTEGER, accumulation, No data) • Total_Files_Sent (INTEGER, accumulation, No data) • Total_Files_Transferred (INTEGER, accumulation, No data) • Total_Get_Requests (INTEGER, accumulation, No data) • Total_Post_Requests (INTEGER, accumulation, No data) • Total_ISAPI_Extension_Requests (INTEGER, accumulation, No data) • Total_Head_Requests (INTEGER, accumulation, No data) • Total_Other_Request_Methods (INTEGER, accumulation, No data) • Total_Logon_Attempts (INTEGER, accumulation, No data) • Total_Anonymous_Users (INTEGER, accumulation, No data)	Resources WINPWSCR WINPWSCA WINPWSFR WINPWSFS WINPWSFT WINPWSGR WINPWSOR WINPWSER WINPWSHR WINPWSRM WINPWSLA WINPWSAU

Table 115. Identifiers and resources that are collected from Tivoli Data Warehouse by the Windows OS agent. (continued)

Tivoli Monitoring agent name	Tivoli Monitoring attribute group	Database table and column names	Resource description and identifier name in SmartCloud Cost Management
	Web Service (cont.)	• Total_NonAnonymous_Users (INTEGER, accumulation, No data) • Total_Delete_Requests (INTEGER, accumulation, No data) • Total_Method_Requests (INTEGER, accumulation, No data) • Total_Put_Requests (INTEGER, accumulation, No data) • Total_Trace_Requests (INTEGER, accumulation, No data) • TOTAAIOREQ (Total Allowed Async I/O Requests) (INTEGER, accumulation, No data) • TOTRAIOREQ (Total Rejected Async I/O Requests) (INTEGER, accumulation, No data) • TOTBAIOREQ (Total Blocked Async I/O Requests) INTEGER, accumulation, No data) IDs • System_Name • Web_Site	WINPWSNU WINPWSDR WINPWSMR WINPWSPR WINPWSTR WINPWSAI WINPWSRI WINPWSBI IDs system_name web_site agent = “windows_os” attribute_group = “Web Service”

Universal data collector

A universal data collector is available for applications that do not have a specific SmartCloud Cost Management data collector.

Setting up the Universal data collector

The Universal data collector uses the Integrator program to convert data in any input usage file into a CSR or CSR+ file. The Integrator program is run from a job file and uses the common XML architecture used for all data collection in addition to elements that are specific to Integrator.

SmartCloud Cost Management includes a sample job file, `SampleUniversal.xml`, that you can modify and use to process any Advanced Accounting usage log. After you have modified and optionally renamed the `Universal.xml` file, move the file to the *<SCCM_install_dir>*`\jobfiles` directory.

Virtual I/O Server data collector

The Virtual I/O Server includes several IBM Tivoli agents and clients, including the ITUAM agent. The ITUAM agent is packaged with the Virtual I/O Server and is installed when the Virtual I/O Server is installed. If the ITUAM agent is run on the Virtual I/O Server, the agent produces a usage log that can be transferred to the ITUAM server on Windows for processing. For more information about the Virtual I/O Server and the ITUAM agent, including how to configure and start the agent, refer to the System p® Advanced Power Virtualization Operations Guide.

Identifiers and resources defined by the Virtual I/O Server data collector

The Virtual I/O Server data collector defines the identifiers and resources described in this topic.

On the Virtual I/O Server, the ITUAM agent gathers data from a Virtual I/O Server data file or files and produces a log file or files. By default, the ITUAM agent creates a daily log file that contains Virtual I/O Server interval records (record type 10). If you want to create separate log files containing any of the record types 1–8 shown in the following list, the UNIX or Linux padmin user must modify the AACCT_TRANS_IDS variable value in the Configuration Parameter file (A_config.par) file on the Virtual I/O Server.

The ITUAM agent creates a separate log file for each of the supported record types in the data files. The record types are:

- 1 Process records
- 4 System processor and memory interval record
- 6 File system activity interval record
- 7 Network interface I/O interval record
- 8 Disk I/O Interval record
- 10 Virtual I/O Server interval record (this is the default record collected)

The following shows the data within the records that is defined as chargeback identifiers and resource rate codes by record type. The rate codes assigned to the resources are preloaded in the Rate table.

Process Data Collected (Record Type 1)

Identifiers

- Feed (this is passed from System i® collector job file)

- AccountingCode
- JobName
- UserName
- ProgramName
- Profile

Resources

- AAID0101 (AIX Process Interval Count)
- AAID0102 (AIX Process Elapsed Time [seconds])
- AAID0103 (AIX Process Elapsed Thread Time [seconds])
- AAID0104 (AIX Process CPU Time [seconds])
- AAID0105 (AIX Elapsed Page Seconds Disk Pages)
- AAID0106 (AIX Elapsed Page Seconds Real Pages)
- AAID0107 (AIX Elapsed Page Seconds Virtual Memory)
- AAID0108 (AIX Process Local File I/O [MB])
- AAID0109 (AIX Process Other File I/O [MB])
- AAID0110 (AIX Process Local Sockets I/O [MB])
- AAID0111 (AIX Process Remote Sockets I/O [MB])

System Data Collected (Record Type 4)

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour

Resources

- AAID0401 (AIX System Number of CPUs [interval aggregate])
- AAID0402 (AIX System Entitled Capacity [interval aggregate])
- AAID0403 (AIX System Pad Length [interval aggregate])
- AAID0404 (AIX System Idle Time [seconds])
- AAID0405 (AIX System User Process Time [seconds])
- AAID0406 (AIX System Interupt Time [seconds])
- AAID0407 (AIX System Memory Size MB [interval aggregate])
- AAID0408 (AIX System Large Page Pool [MB])
- AAID0409 (AIX System Large Page Pool [MB in-use])
- AAID0410 (AIX System Pages In)
- AAID0411 (AIX System Pages Out)
- AAID0412 (AIX System Number Start I/O)
- AAID0413 (AIX System Number Page Steals)

File System Data Collected (Record Type 6)

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- FS_TYPE (file system type)
- Device
- MOUNT_PT (mount point)

Resources

- AAID0601 (AIX FS Bytes Transferred [MB])
- AAID0602 (AIX FS Read/Write Requests)
- AAID0603 (AIX FS Number Opens)
- AAID0604 (AIX FS Number Creates)
- AAID0605 (AIX FS Number Locks)

Network Data Collected (Record Type 7)

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- Interface

Resources

- AAID0701 (AIX Network Number I/O)
- AAID0702 (Network Bytes Transferred [MB])

Disk Data Collected (Record Type 8)

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour

- DISKNAME

Resources

- AAID0801 (AIX Disk Transfers)
- AAID0802 (AIX Disk Block Reads)
- AAID0803 (AIX Disk Block Writes)
- AAID0804 (AIX Disk Transfer Block Size [interval])

Virtual I/O Server Data Collected (Record Type 10)

Identifiers

- SYSTEM_ID
- Sysid
- Sysmodel
- Partition_Name
- Partition_Number
- Trans_Project
- Sub_Project_Id
- Hour
- SERPARNO (server partition number)
- SERUNID (server unit number)
- DLUNID (device logical unit ID)

Resources

- AAID1001 (Virtual I/O Server Bytes In)
- AAID1002 (Virtual I/O Server Bytes Out)

Transferring usage logs from the Virtual I/O Server to the SmartCloud Cost Management application server for processing

To transfer the usage logs from the Virtual I/O Server for processing on the SmartCloud Cost Management application server, you can do either of the following:

- "Pull" the logs from Virtual I/O Server to the SmartCloud Cost Management application server. This method requires that you set up a job file on your Windows platform to pull the file to the SmartCloud Cost Management server for processing. This is the recommended method because the transfer process, including any failed transfers, is included in the nightly job log.
- "Push" the logs from the Virtual I/O Server to the SmartCloud Cost Management application server. This method requires that the UNIX or Linux user modify the `A_config.par` file on the Virtual I/O Server.

Pulling usage logs to the SmartCloud Cost Management application server

The following are the steps required to pull the usage logs from the Virtual I/O Server the SmartCloud Cost Management application server. By default, the usage logs are pulled from and to the following locations:

- On the Virtual I/O Server, the usage logs are in `<SCCM_install_dir>/collectors/unix/aacctn_<date>.txt`, where `n` is the record type.

- On the SmartCloud Cost Management application server, the usage logs are sent to <*SCCM_install_dir*>/logs/collectors/AACCT_n/<feed>, where n specifies the record type and feed specifies the log file source.

Example

In this example, usage log containing type 10 records from a Virtual I/O Server with the name vioray are transferred to the following folder on the SmartCloud Cost Management application server on the Windows platform: <*SCCM_install_dir*>/logs/collectors/AACCT_10/vioray.

The collectors and vioray (feed) directories are defined by the RPDParameters parameter in the transfer job file. The ACCT_10 folder is defined by the manifest.

Creating a transfer job file to collect the usage log

SmartCloud Cost Management includes a sample job file, SampleSecureGetVIOS.xml, that you can modify and use to transfer the Virtual I/O Server usage log to the SmartCloud Cost Management application server. After you have modified and optionally renamed the SampleSecureGetVIOS.xml file, move the file to the <*SCCM_install_dir*>/jobfiles directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<Jobs xmlns="http://www.ibm.com/TUAMJobs.xsd">
    <Job active="true" dataSourceId=""
         description="Get CSR files from ITUAM UNIX/Linux Collector Agent"
         id="SecureGetVIOS"
         joblogShowStepOutput="true"
         joblogShowStepParameters="true"
         joblogWriteToTextFile="true"
         joblogWriteToXMLFile="true"
         processPriorityClass="Low"
         smtpFrom="ITUAM@ITUAMCustomerCompany.com"
         smtpSendJobLog="false"
         smtpServer="mail.ITUAMCustomerCompany.com"
         smtpTo="John.ITUAMUser@ITUAMCustomerCompany.com"
         stopOnProcessFailure="false">

        <Process active="true"
                 description="Get Advanced Accounting Files from VIOS"
                 id="SecureGetVIOS"
                 joblogShowStepOutput="true"
                 joblogShowStepParameters="true">

            <Steps stopOnStepFailure="true">
                <Step active="true"
                      description="Collect Advanced Accounting Files for client1"
                      id="SecureGetVIOS_client1"
                      programName="rpd"
                      programType="java"
                      type="ConvertToCSR">

                    <Parameters>
                        <Parameter/>
                        <Parameter Host="192.168.0.109"/>
                        <Parameter UserId="username"/>
                        <Parameter Password="password"/>
                        <Parameter Manifest="SecureGetVIOS.xml"/>
                        <Parameter RPDParameters="client_CS_path=&quot;/opt/IBM/tivoli/ituam/
collectors/Unix/CS_input_source&quot;;CollectorLog_dir=&quot;%CIMSInstallLocation%/logs/
collectors&quot;;LogDate=%LogDate%;client_name=client1;"/>
                        <Parameter Verbose="true"/>
                        <Parameter SourcePath="&quot;%CIMSInstallLocation%&quot;/collectors/unixlinux/"/>
                    </Parameters>
                </Step>
            </Steps>
        </Process>

    </Job>
</Jobs>
```

The parameters shown in the following table are specific to the SampleSecureGetVIOS.xml file. These parameters are required unless specified as optional.

Table 116. SampleSecureGetVIOS.xml Parameters

Parameter	Description/Values
Parameter Host	The IP address or DNS name of the Virtual I/O Server from which you want to transfer the usage logs.
Parameter UserId Parameter Password	The user ID and password to connect to the Virtual I/O Server.
Parameter Manifest	The manifest file. The manifest is an XML file that specifies the usage logs that you want to collect from the Virtual I/O Server. Place the manifest in the <*SCCM_install_dir*>collectors/unixlinux directory and define the path in the SourcePath parameter.
RPDParameters (Optional)	These parameters are used to define the from and to locations for transferring the usage log files from the Virtual I/O Server to the SmartCloud Cost Management application server; the date for the usage log or logs that you want to collect; and the name of the Virtual I/O Server from which you are transferring the files. These are the same parameters that are in the Deployment element of the sample manifest. The parameter values that you define her override the default Deployment parameter values in the manifest file. It is recommended that you do not change the default values in the manifest file. The following describes each of these parameters. ■ client_CS_path—The path to the usage logs on the Virtual I/O Server. Do not change this parameter. ■ CollectorLog_dir—The path to the CollectorLogs folder on the SmartCloud Cost Management application server. Do not change this parameter unless you want to store the usage logs in a folder other than CollectorLogs. ■ LogDate—The log date specifies the date for the log file that you want to collect. ■ client_name—The name of the Virtual I/O Server from which you are collecting the usage logs. This name is used to create the feed subfolder in the path defined by the CollectorLog_dir parameter.
Verbose (Optional)	The value -verbose specifies that the addition information is included in the job log file or 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.
SourcePath	The full path to the <*SCCM_install_dir*>collectors/unixlinux directory on the SmartCloud Cost Management application server. This folder must contain the manifest file.

Creating a manifest

SmartCloud Cost Management includes a sample manifest file, SampleSecureGetVIOSManifest.xml, that you can modify and use for your organization. After you have modified and optionally renamed the SampleSecureGetVIOSManifest.xml file, move the file to the <*SCCM_install_dir*>/collectors/unixlinux directory

Because the ITUAM agent produces usage logs that contain record type 10 by default, the Action elements in the manifest for usage logs containing all other

record types are commented. To specify whether a specific usage log is collected, simply comment or uncomment the `Action` element for that log.

With the exception of commenting or uncommenting the usage logs that are collected as needed, it is recommended that you do not change the default values in the manifest file. The parameters in the `Deployment` element are set by the RPDParameters attributes in the transfer job file. In addition, the `Action` element attribute and parameter values include macro capability so that the pre-defined strings and environment strings that are expanded at run time. You should not change these values.

```
<?xml version="1.0"?>
<ProductDeployment name = "UNIX/Linux"
                   description = "Secure Get SmartCloud Cost Management UNIX/LINUX Data Collector CSR Files">
    <Deployments>
        <Deployment name="collection" targetOS="AIX" deploymentType="install">
   <Parameters>
   <Parameter name="client_CS_path"    required="no"  defaultValue="/opt/IBM/tuam/collectors/unix/
           CS_input_source"/>
   <Parameter name="CollectorLogs_dir" required="no"  defaultValue="C:/Program Files/IBM/ITUAM/logs/
           collectors"/>
   <Parameter name="client_name"    required="yes"/>
   <Parameter name="LogDate"     required="yes"/>
   </Parameters>

            <Actions>

<!--
              <Action name="step_AACCT_1_%client_name%" displayMessage="Getting nightly AACCT_1 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_1//%client_name%"/>
                   <Parameter name="remotefilename"    value="%client_CS_path%/aacct1_%LogDate%.txt"/>
          </Parameters>
                </Action>
-->
<!--
              <Action name="step_AACCT_4_%client_name%" displayMessage="Getting nightly AACCT_4 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_4//%client_name%"/>
                    <Parameter name="remotefilename"    value="%client_CS_path%/aacct4_%LogDate%.txt"/>
          </Parameters>
                </Action>
-->
<!--
              <Action name="step_AACCT_6_%client_name%" displayMessage="Getting nightly AACCT_6 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_6//%client_name%"/>
                    <Parameter name="remotefilename"    value="%client_CS_path%/aacct6_%LogDate%.txt"/>
          </Parameters>
                </Action>
-->
<!--
              <Action name="step_AACCT_7_%client_name%" displayMessage="Getting nightly AACCT_7 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_7//%client_name%"/>
                    <Parameter name="remotefilename"    value="%client_CS_path%/aacct7_%LogDate%.txt"/>
          </Parameters>
                </Action>
-->
<!--
              <Action name="step_AACCT_8_%client_name%" displayMessage="Getting nightly AACCT_8 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_8//%client_name%"/>
                    <Parameter name="remotefilename"    value="%client_CS_path%/aacct8_%LogDate%.txt"/>
          </Parameters>
                </Action>
-->
              <Action name="step_AACCT_10_%client_name%" displayMessage="Getting nightly AACCT_10 file
                  for %client_name%" actionType="FileGet">
                  <Parameters>
                    <Parameter name="localpath" value="%CollectorLogs_dir%//AACCT_10//%client_name%"/>
                    <Parameter name="remotefilename"    value="%client_CS_path%/aacct10_%LogDate%.txt"/>
          </Parameters>
                </Action>

            </Actions>

        </Deployment>
```

```
    </Deployments>
</ProductDeployment>
```

Running the transfer job file

To run the transfer job file from the command prompt, use the following command:

`startJobRunner.sh <transfer file name>.xml`

Where <*transfer file name*> is the name that you give to the transfer job file.

To run the transfer job file from Administration Console:

1. In Administration Console, click **Task Management** > **Job Runner** > **Job Files**.
2. On the Job Runner Job Maintenance page, click the **View popup menu** icon for the job file that you want to run and then click **Run Job**.

Note: Make sure that the transfer file is in the <*SCCM_install_dir*>/jobfiles directory so that it will be displayed on the Job Runner Job Maintenance page.

Setting up Virtual I/O Server data collection

This topic provides information about setting up Virtual I/O Server data collection.

Note: The AIX Advance Accounting and the Virtual I/O Server data collectors use the same sample job file.

The Virtual I/O Server data collector uses the Integrator program to convert data in the Virtual I/O Server usage log file into a CSR or CSR+ file. The Integrator program is run from a job file and uses the common XML architecture used for all data collection in addition to elements that are specific to Integrator.

SmartCloud Cost Management includes a sample job file, SampleAIXAA.xml, that you can modify and use to process the Virtual I/O Server usage log. After you have modified and optionally renamed the SampleAIXAA.xml file, move the file to the <*SCCM_install_dir*>\jobfiles directory.

VMware data collector

The VMware collector collects data from VMware VirtualCenter Server 2.5, vCenter 4.x and 5.0.

Configuring VMware server systems to enable SmartCloud Cost Management VMware collection

This topic describes how to configure VMware server systems to enable SmartCloud Cost Management VMware collection.

About this task

To enable VMware data collection, a VMware server system requires the following configuration:

Procedure

1. The SmartCloud Cost Management VMware collector uses the VMware Infrastructure API (VI API) to gather metrics from VMware Infrastructure (VI) servers i.e., VC. This VI API is exposed as a web service on VI servers. The Web Service therefore needs to be installed and running on the VI server from

which you want to gather metrics. The default installation includes the Web Service. However, if a custom installation was performed then the Web Service option may not be installed. To confirm that the Web Service API is installed and running:

a. Using a Web browser from your SmartCloud Cost Management server, connect using the URL of the server name or IP address.
b. If you are required, accept the webpage certificate. Upon success a **VMware Virtual Center** welcome screen is displayed.
c. If web access is enabled, login using the **Log in to Web Access** link with your server authentication information. If web access is not enabled, then you can check by referencing the "Managed Object Base" (MOB) at the URL of the server name or IP address/mob, for example: `https://<IP_Address>/MOB`.

2. Statistics Collection Level 3 includes all metrics (including device metrics) for all counter groups (average, summation, and latest) rollup types. To ensure that the SmartCloud Cost Management collector collects all the specified metrics, then it is recommended to use this level setting. Refer to VMware documentation or the related topic about the VMware usage metrics collected for details about what performance counters are persisted for each statistics level. The level can be checked using the VMware Infrastructure Client (VI Client):
   a. Log in.
   b. Click **Administration** > **VirtualCenter Management Server Configuration**.
   c. Select **Statistics** in the dialog and change the **Statistics Level** for the **Statistics Intervals** to level 3 if necessary. The level only needs to be set to 3 for the required collection interval.

Related reference:

"VMware usage metrics collected"
This topic describes the usage metrics collected by the VMware collector.

VMware usage metrics collected

This topic describes the usage metrics collected by the VMware collector.

The following tables shows the VMware usasge metrics collected by VMware group name.

Table 117. cpu

Counter Name	Category/Value	Statistics Level
`usage`	**Unit**: percentage precision to 1/100 percentage point. 1 = 0.01%. A value between 0 and 10000 **Description**: CPU usage as a percentage over the interval of collection **Statistic type**: rate **Rollup type**: average, maximum and minimum **Entity**: host, virtual machine	Level 1

Table 117. cpu (continued)

Counter Name	Category/Value	Statistics Level
usagemhz	**Unit**: MHz **Description**: CPU usage in MHz over the interval of collection **Statistic type**: rate **Rollup type**: average, maximum and minimum **Entity**: host, virtual machine, compute resources and resource pools	Level 1
system	**Unit**: millisecond **Description**: CPU time spent on system processes **Statistic type**: delta - A value that reports the change that happened during the last sample period **Rollup type**: a **Entity**: virtual machine (per cpu instance only)	Level 3
wait	**Unit**: millisecond **Description**: CPU time spent on wait state **Statistic type**: delta **Rollup type**: summation **Entity**: virtual machine (per cpu instance only)	Level 3
ready	**Unit**: millisecond **Description**: CPU time spent on ready state **Statistic type**: delta **Rollup type**: summation **Entity**: virtual machine (per cpu instance only)	Level 1

Table 117. cpu (continued)

Counter Name	Category/Value	Statistics Level
extra	**Unit**: millisecond **Description**: CPU time that is extra **Statistic type**: delta **Rollup type**: summation **Entity**: virtual machine (per cpu instance only)	Level 3
used	**Unit**: millisecond **Description**: CPU time that is used **Statistic type**: delta **Rollup type**: summation **Entity**: virtual machine (per cpu instance only)	Level 3
guaranteed	**Unit**: millisecond **Description**: CPU time that is guaranteed for the virtual machine **Statistic type**: delta **Rollup type**: summation **Entity**: virtual machine (per cpu instance only)	Level 3

Table 118. net

Counter Name	Category/Value	Statistics Level
packetRx	**Unit**: number **Description**: Number of packets received in the performance interval **Statistic type**: delta **Rollup type**: summation **Entity**: host, virtual machine (per net instance only)	Level 3

Table 118. net (continued)

Counter Name	Category/Value	Statistics Level
packetTx	**Unit**: number **Description**: Number of packets transmitted in the performance interval **Statistic type**: delta **Rollup type**: summation **Entity**: host, virtual machine (per net instance only)	Level 3

Table 119. disk

Counter Name	Category/Value	Statistics Level
numberRead	**Unit**: number **Description**: Number of times data was read from the disk in the defined interval **Statistic type**: delta **Rollup type**: summation **Entity**: host, virtual machine (per net instance only)	Level 3
numberWrite	**Unit**: number **Description**: Number of times data was written to the disk in the defined interval **Statistic type**: delta **Rollup type**: summation **Entity**: host, virtual machine (per net instance only)	Level 3

Table 120. mem

Counter Name	Category/Value	Statistics Level
active	**Unit**: kilobytes **Description**: Amount memory that is actively used **Statistic type**: absolute **Rollup type**: average, maximum and minimum **Entity**: host, virtual machine, compute resources, resource pools	Level 2

Table 120. mem (continued)

Counter Name	Category/Value	Statistics Level
granted	**Unit**: kilobytes **Description**: Amount of memory available for use **Statistic type**: absolute **Rollup type**: average, maximum and minimum **Entity**: host, virtual machine, compute resources, resource pools	Level 2

Note: The use of higher level statistics, for example, Level 3 in a Virtual Center/vCenter requires that the Virtual Center/vCenter database is capable of handling large data volumes and proper maintenance is performed on the database.

Note: The statistic levels stated in this topic are correct as of the vCenter 5.0 documentation.

Related tasks:

"Configuring VMware server systems to enable SmartCloud Cost Management VMware collection" on page 333
This topic describes how to configure VMware server systems to enable SmartCloud Cost Management VMware collection.

Identifiers and resources defined by the VMware collector

The VMware data collector defines the identifiers and resources described in this topic.

By default, the following data collected by the VMware collector is defined as chargeback identifiers and resource rate codes in the `SampleVMWare.xml` job file. The rate codes assigned to the resources are pre-loaded in the Rate table.

Identifiers

- Feed (defined in the VMware collector job file)
- DataCenterName (the name of the data center)
- HostName (the name of the host server)
- HostName1 (the name of a host server in a cluster) [1]
- ResourcePoolName1 (the name of the resource pool) [2]
- VMName (the name of the virtual machine)
- VMDescription (a description for the virtual machine)
- VMGuestOSName (the full name of the guest operating system for the virtual machine)

1. Depending on how many hosts are added to the cluster, you can have multiple host name identifiers (that is, HostName1, HostName2, HostName3, and so on).
2. Depending on the number of levels of resource pools that are configured, you can have multiple resource pool identifiers (that is, ResourcePoolName1, ResourcePoolName2, ResourcePoolName3, and so on).

- VMInstance (the instance of the device)
- DNSName (the DNS name of the host or the virtual machine)

Resource Rate Codes

- VMCPUSYS (CPU Time Spent on System Processes)
- VMCPUWAT (CPU Time Spent on Wait State)
- VMCPURDY (CPU Time Spent on Ready State)
- VMCPUEXT (CPU Time That is Extra) [3]
- VMCPUPCT (CPU Usage as a Percentage Over the Interval of Collection) [4]
- VMCPUMHZ (CPU Usage in MHz Over the Interval of Collection)
- VMCPUUSE (VMware CPU Usage)
- VMCPUGUA (VMware CPU Usage Guaranteed) [3]
- VMNETREC (VMware Network Packets Received)
- VMNETTRN (VMware Network Packets Transferred)
- VMDSKWRI (VMware Number of Disk Writes)
- VMDSKRED (VMware Number of Disk Reads)
- VMMEMAVL (Amount of Memory in Kilobytes That is Available for Use)[4]
- VMMEMUSD (Amount of Memory in Kilobytes That is Actively Used) [4]
- RPCPULMT (Resource Pool Max Limit Amount of CPU in MHz)
- RPCPURES (Resource Pool Amount of CPU in MHz That Is Guaranteed Available)
- RPCPUEXP (Resource Pool CPU Reservation Is Fixed (1) or Expandable (2))[5]
- RPMEMLMT (Resource Pool Max Limit Amount of Memory in MB)
- RPMEMRES (Resource Pool Amount of Memory in MB That Is Guaranteed Available)
- RPMEMEXP (Resource Pool Memory Reservation Is Fixed (1) or Expandable (2))[5]
- VMCPURES (Configured CPU Reservation of the virtual machine in MHz)
- VMMEMRES (Configured Memory Reservation of the virtual machine in MB)
- VMMEMSIZ (Memory Size of the virtual machine in MB)
- VMNUMCPU (Number of processors in the virtual machine)
- VMDSCP1 (Total Capacity of the Disk Allocated to the virtual machine in KB)[6]

3. Metric maybe deprecated depending on the version of vCenter. Available up to and including ESX v3.5.0.

4. It is recommended that this data is written to the resource utilization table. For more information, see the *DBLoad specific parameter attributes* topic in the *Reference guide*.

5. If value is 1 then there is no fixed limit.

6. Multiple disk capacity rate codes can exist (that is, VMDSCP1, VMDSCP2, VMDSCP3, and so on).

Setting up SmartCloud Cost Management for VMware data collection

This topic provides details on setting up SmartCloud Cost Management for VMware data collection.

Configuring SmartCloud Cost Management for the HTTPS protocol

The VMware Infrastructure (VI) web service uses HTTPS as its communication protocol. The HTTPS protocol uses SSL certificates. For a client (e.g., the SmartCloud Cost Management VMware collector) to connect to the web service it needs the self-generated SSL certificate of the VI server with which it wants to communicate.

For VC/vCenter installed on Windows 2003, the self-generated SSL certificate can be located at: `C:\Documents and Settings\All Users\Application Data\VMware\VMware VirtualCenter\SSL\rui.crt`. For VC\vCenter installed on Windows 2008, the self-generated SSL certificate can be located at: `C:\ProgramData\VMWARE\VMWARE VirtualCenter\SSL\rui.crt`. Consult the VMware documentation for more details on other systems. The certificate needs to be imported into a truststore on the SmartCloud Cost Management server.

To create this truststore on the Windows platform:

1. Open a command prompt.
2. Create the `C:\VMWare-Certs` directory.
3. Make sure the Java SDK tools are in your path.
4. Change to the `C:\VMWare-Certs` directory.
5. Import a certificate.
   a. Enter the keytool command:

      ```
      keytool -import -file <certificate-filename> -alias <server-name> -keystore vmware.keystore
      ```

      For example:

      ```
      keytool -import -file c:\vmware\rui.crt -alias VMwareServer1 -keystore c:\VMware-Certs
      \vmware.truststore
      ```

   b. Enter a password for the truststore. (Note the password. You will must type the password on the Web Service Data Source Maintenance page in Administration Console.)
   c. Enter yes to import the certificate.

To create this truststore on a Linux or UNIX platform:

1. Open a shell
2. Create the `~/VMware-Certs` directory.
3. Make sure the Java SDK tools are in your path.
4. Change to the `~/VMware-Certs` directory.
5. Import a certificate.
   a. Enter the keytool command:

      ```
      keytool -import -file <certificate-filename> -alias <server-name> -keystore vmware.keystore
      ```

      For example:

      ```
      keytool -import -file ~/VMware/rui.crt -alias VMwareServer1 -keystore ~/VMware-Certs
      /vmware.truststore
      ```

   b. Enter a password for the truststore. (Note the password. You will must type the password on the Web Service Data Source Maintenance page in Administration Console.)
   c. Enter yes to import the certificate.

Creating a web service data source

You must create a data source in the Administration Console that points to the VMware VirtualCenter or VMware Server Web service. The data source is referenced in the VMware collector job runner file. To create the data source in Administration Console:

1. In Administration Console, click **System Configuration** > **Data Sources** and select **Web service** as the **Data Source Type**.
2. Click **Create Data Source**.
3. Complete the following:

   Data Source Name
   Type the name that you want to assign to the data source.

   Note: The following are invalid characters for a data source name: "/", "\","","":","?","<",">",".","|",".".

   User Name
   Type the Web service user ID.

   Password
   Type the Web service password.

   URL Type the Web service URL as follows.

   Note: The port is not required in the URL unless you are using a port other than 80 for HTTP and 443 for HTTPS.

   `http://<Server Name>:port`

   Or

   `https://<Server Name>:port`

   To determine the port number:

   - Using the Virtual Center Infrastructure client, click **Administration** on the menu bar.
   - Click **VirtualCenter Management Server Configuration**.
   - In the **VirtualCenter Management Server Configuration** dialog box, click **Web Service**. The ports are shown in the **HTTP:** and **HTTPS:** boxes.

   Web Service Type
   Select **VMware** as the web service type.

   Keystore File
   The Keystore or truststore file contains the vCenter or REST server certificate that is used for authentication during the secure connection between the collector and the vCenter web service. The password is used to access the file. Enter a valid path to the file.

   Keystore Password
   Type the Keystore or truststore password.
4. Click **Create** to save the data source information. The new data source name is displayed in the **Data Source Name** menu.

   Note: When the data source information is saved, the connection to the vCenter or REST server is verified. You should see a message at the top of the screen indicating that the connection was successful.

Editing the sample job file

The VMware data collector uses the Integrator program to convert data collected by the collector into a CSR or CSR+ file. The Integrator program is run from a job file and uses the common XML architecture used for all data collection in addition to elements that are specific to Integrator.

SmartCloud Cost Management includes a sample job file called `SampleVMWare.xml` that is used for data collection. This XML file can be modified and used to process the VMware data. After you have modified and optionally renamed the file, move the file to the *<SCCM_install_dir>*`\jobfiles` directory.

The sample job files contain the following parameters that are specific to VMware collection.

```
<Input name="CollectorInput" active="true">
  <Collector name="VMWARE">
  <WebService dataSourceName="VMWareCollector"/>
  <Interval id="1800"/>
 </Collector>
 <Parameters>
  <Parameter name="aggregateDaily" value="true" DataType="Bool"/>
 </Parameters>
```

These parameters are described in the following table.

Table 121. VMware Parameters

Parameter	Description/Values
WebService-dateSourceName	The name of the SmartCloud Cost Management data source that points to the VMware VirtualCenter Server or ESX Server.
WebService-trustStore	The full path name of the truststore. (This parameter is now obsolete and overwritten by the web service data source configuration item.)
Interval-id	The VI server needs to be configured so that metrics are collected and written to the VI server database. It can be configured for different intervals, for example, 300 means that data will be collected for 5 minute time intervals, 1800 for 30 minute time intervals, 7200 for 2 hour interval. The Interval-id parameter selects which of the intervals that the metrics should be gathered from the VI server database. The Interval-id can be set to, for example, 300, 1800 or 7200. corresponding to the intervals configured in the VI server. If the Interval-id used is not configured in the VI server, then the interval is unknown and the metrics therefore cannot be gathered.
`aggregateDaily`	Aggregates all interval metrics to daily if set to true. True by default if not set.

The `SampleVMWareReserved.xml` job file shows how to calculate resources, which can be used to charge different rates for usage within a reservation and usage that exceeded it (CPU or memory).

Mapping of VMware Identifiers and resources to VMware Infrastructure Equivalents

This topic describes how the identifier and resources defined by the VMware data collector map to performance usage counters and system properties (as retrieved by VI API calls) of a VMware Infrastructure (VI) server.

Table 122. Identifiers

Name	VI Equivalent
Feed	N/A
DataCenterName	ManagedObjectReference::getPropSet().getVal()
HostName	HostConfigSummary::getName()
ResourcePoolName1	ResourcePool::getName() or “default” if name is null
VMName	VirtualMachineConfigSummary::getName()
VMDescription	VirtualMachineConfigSummary::getAnnotation()
VMGuestOSName	VirtualMachineConfigInfo::getGuestFullName()
VMInstance	PerfMetricIntSeries::getId()::getInstance()
DNSName	GuestInfo::getHostName()

Table 123. Resources

Resources	VI Equivalent
VMCPUPCT	cpu:usage
VMCPUMHZ	cpu:usagemhz
VMCPUWAT	cpu:wait
VMCPURDY	cpu:ready
VMCPUEXT	cpu:extra
VMCPUUSE	cpu:used
VMCPUSYS	cpu:system
VMCPUGUA	cpu:guaranteed
VMMEMUSD	mem:active
VMMEMAVL	mem:granted
VMNETREC	net:packetsRx
VMNETTRN	net:packetsTx
VMDSKRED	disk:numberRead
VMDSKWRI	disk:numberWrite
RPCPULMT	ResourceAllocationInfo::getLimit()
RPCPURES	ResourceAllocationInfo:: getReservation()
RPCPUEXP	ResourceAllocationInfo:: getExpandableReservation()
RPMEMLMT	ResourceAllocationInfo::getLimit()
RPMEMRES	ResourceAllocationInfo:: getReservation()
RPMEMEXP	ResourceAllocationInfo:: getExpandableReservation()
VMCPURES	ResourceConfigSpec::getCpuAllocation::getReservation()
VMMEMRES	ResourceConfigSpec::getMemoryAllocation::getReservation()
VMMEMSIZ	VirtualMachineConfigSummary:: getMemorySizeMB()

Table 123. Resources (continued)

Resources	VI Equivalent
VMNUMCPU	VirtualMachineConfigSummary:: getNumCpu()
VMDSCP1	VirtualDevice:: getCapacityinKB()

Vmstat data collector

Vmstat is a command available on all UNIX/Linux platforms which provides the user with system level CPU and memory, utilization and capacity information.

Vmstat reports information in time intervals as requested by the user. For example, the user may request vmstat to report 10 5-minute intervals of utilization and capacity information.

```
# vmstat 300 10
```

The SmartCloud Cost Management vmstat collector provides a script which is scheduled in root's crontab to generate vmstat reports and transfer the reports to the SmartCloud Cost Management Server on a daily basis for processing. The vmstat reports are processed using a SmartCloud Cost Management jobrunner job file. SmartCloud Cost Management Server provides the following files to implement the SmartCloud Cost Management vmstat Collector.

```
.../collectors/vmstat/vmstat_collect.template
.../collectors/vmstat/tuam_unpack_vmstat
.../collectors/vmstat/vmstatDeploymentManifest_linux.xml
.../collectors/vmstat/vmstatDeploymentManifest_hp.xml
.../collectors/vmstat/vmstatDeploymentManifest_aix.xml
.../collectors/vmstat/vmstatDeploymentManifest_solaris.xml

.../samples/jobfiles/SampleDeployVmstatCollector.xml
.../samples/jobfiles/SampleVmstat.xml
```

Identifiers and resources collected by vmstat collector

Vmstat reports are converted to CSR record format by an Integrator stage in the vmstat jobfile.

The following identifiers and resources are included in the CSR file.

DITA

Identifiers:

Table 124. Vmstat Identifiers

Identifier	Description
SYSTEM_ID	Servername
IP_ADDR	IP address of server
OP_SYS	Operating system of server
INT_START	Interval start time YYYYMMDDHHMMSS
INT_END	Interval end time YYYYMMDDHHMMSS

Resources:

Table 125. Vmstat Resources

Resource	Description
VSNUMCPU	Number physical CPUs
VSMEMSIZ	Memory Size (KB)
VSMEMFRE	Memory Free (KB)
VSUCPUPT	Percent User CPU
VSSCPUPT	Percent System CPU
VSICPUPT	Percent Idle CPU
VSWAIOPT	Percent Wait I/O
VSUCPUUS	User CPU used (seconds)
VSSCPUUS	System CPU used (seconds)
VSICPUUS	Idle CPU Time (seconds)
VSWAIOUS	Wait I/O Time (seconds)

It is recommended that this data is written to the resource utilization table. For more information on how to do this, see *DBLoad* topic in the *Reference Guide*.

Deploying the vmstat collector

The SmartCloud Cost Management vmstat Collector can be deployed in one of two ways. First, using the RXA capability built into SmartCloud Cost Management Server. This method allows the user to install the vmstat collector to the target server from the SmartCloud Cost Management Server. A second method would be to manually copy the vmstat installation files to the target server and run the tuam_unpack_vmstat installation script.

Regardless of the method used to install, the collector requires that the SmartCloud Cost Management Server is running an FTP service to receive the nightly vmstat reports from the target servers. Where SmartCloud Cost Management server is running on Windows, you will need to be sure that:

- The IIS FTP service is installed.
- A virtual directory has been defined to the SmartCloud Cost Management Collector Logs folder.
- The FTP service can be configured to allow either anonymous or user/password access, but the accessing account must have read/write access to the SmartCloud Cost Management Collector Logs folder.

Installing the SmartCloud Cost Management vmstat Collector from the SmartCloud Cost Management Server:

Installing the vmstat Collector from the SmartCloud Cost Management Server is accomplished using the `SampleDeployVmstatCollector.xml` jobfile.

Before you begin

Several parameter settings need to be edited prior to running the jobfile.

- **Host**: The hostname or IP address of the target server.
- **UserId**: Must be set to "root".
- **Password**: Password for root on the target server.
- **Manifest**: Manifest file for the OS type of the target server

- **RPDParameters**: See note in the example.

Example

Example from `SampleDeployVmstatCollector.xml` jobfile:

Note: Example lines that are too long are broken up. The breakpoint is indicated using a backslash "\".

```
                <!-- SUPPLY hostname OF TARGET PLATFORM/-->
                <Parameter Host             = "TARGET_PLATFORM"/>

                <!-- userid must be set to root/-->
                <Parameter UserId           = "root"/>

                <!-- SUPPLY root PASSWORD ON TARGET PLATFORM/-->
                <Parameter Password         = "XXXXXX"/>

                <!--Parameter KeyFilename     = "yourkeyfilename"/-->

                <!-- DEFINE Manifest TO MANIFEST XML FOR TARGET PLATFORM/-->
                <!--Parameter Manifest         = "vmstatDeploymentManifest_linux.xml"/-->
                <!--Parameter Manifest         = "vmstatDeploymentManifest_hp.xml"/-->
                <!--Parameter Manifest         = "vmstatDeploymentManifest_linux.xml"/-->
                <!--Parameter Manifest         = "vmstatDeploymentManifest_solaris.xml"/-->
                <Parameter Manifest         = "vmstatDeploymentManifest_aix.xml"/>

                <!--Parameter Protocol        = "win | ssh"/-->

                <!-- DEFINE INSTALLATION PARAMETERS,
                       path:   must be defined to the directory path where vmstat Collector
                               will be installed on target platform.
                               Parameter Required.
                       server: name or IP address of the SmartCloud Cost Management Server
                               Parameter Required.
                       log_folder: CollectorLog folder on SmartCloud Cost Management Server.
If SmartCloud Cost Management Server is UNIX/Linux
                               platform, log_folder should be defined to the full path ti the
                               Collector Logs folder, example: /opt/ibm/tuam/logs/collectors .
                               If SmartCloud Cost Management Server is Windows platform, log_folder should be set to
                               virtual directory that references the Collector Logs folder.
                               Parameter Required.
                       ftp_user: Account used to access the SmartCloud Cost Management Server for nightly transfer of the
                               vmstat log to the SmartCloud Cost Management Server.  If the log file is to remain on the
                               client, set ftp_user=HOLD. If anonymous ftp access has been
            configured
                               on the SmartCloud Cost Management Server, enter ftp_user=anonymous. This account must have
                               read/write access to the Collector Logs folder.
                               Parameter Required.
                       ftp_key: password used by ftp_user. If ftp_user=anonymous or ftp_user=HOLD,
                               enter any value for this parameter, but enter something,
                               example: ftp_key=XXXX.
                               Parameter Required.
                       add_ip: Indicate if the name of the folder receiving the vmstat files on
                               the SmartCloud Cost Management Server should include the IP address of the client server.
                               example: If log_folder=/opt/ibm/tuam/logs/collectors and the
                               client server  is named "client1", the nightly vmstat file will be
                               placed in
                                   /opt/ibm/tuam/logs/collectors/vmstat/client1
                               If add_ip=Y, the file will be delivered to
                                   /opt/ibm/tuam/logs/collectors/vmstat/client1_<ipaddress>.
                               Default value "N".
                       interval: Number of seconds in vmstat intervals.
                               Default value 300.
                -->
                <Parameter RPDParameters    = "path=/data/tuam/collectors/vmstat;server= \
        9.42.17.133;log_folder=/data/tuam/logs/collectors;ftp_user=ituam;ftp_key= \
        ituam;add_ip=Y;interval=300;"/>
```

Manually installing the vmstat collector:

This topic describes how you can manually install the vmstat collector.

About this task

The SmartCloud Cost Management vmstat Collector can be installed manually by copying the collectors installation files to the target server, placing them in the directory where you want the collector to reside and then executing the collector unpack script.

For example:

1. Suppose SmartCloud Cost Management Server is installed in `<SCCM_install_dir>`.

2. Suppose you want the SmartCloud Cost Management vmstat collector to reside in /opt/ibm/SCCM/collectors/vmstat on the target server.
3. Copy <SCCM_install_dir>/collectors/vmstat_collect.template to /opt/ibm/tuam/collectors/vmstat on the target server.
4. Copy <SCCM_install_dir>/collectors/tuam_unpack_vmstat to/opt/ibm/SCCM/collectors/vmstat on the target server.

Now, on the target server, as root, do the following. (Note: the values supplied are described in the installing topic).

Note: Code lines that are too long are broken up. The breakpoint is indicated using a backslash "/".

```
# cd  / opt/ibm/tuam/collectors/vmstat
# chmod 770 tuam_unpack_vmstat
# ./ tuam_unpack_vmstat  path=/opt/ibm/tuam/collectors/vmstat /
 server=tuamserver  log_folder=collector_log  ftp_user=ituam  /
ftp_key=ituam  add_ip=N  interval=300
```

Following the installation of the vmstat collector:

This topic discusses the actions that must be taken after you have installed the vmstat collector.

The installation places an entry in the root crontab to run the vmstat_collect.sh script once each hour. The script will output an hourly vmstat report to /<SCCM_install_dir>/data/vmstat_YYYYMMDD_HH.txt.

When called in the 23rd hour (11 PM), the script will generate the hourly report and then concatenate the hourly vmstat reports into a single file vmstat_YYYYMMDD.txt. This file is then transferred to the SmartCloud Cost Management Server and placed in <collector_logs>/vmstat/<target_name>. A log of the FTP transfer is written to /<SCCM_install_dir>/data/YYYYMMDD_ftp.log

The daily vmstat reports and ftp logs are retained on the target platform for 10 days.

Script - vmstat_collect.sh

Usage:

```
vmstat_collect.sh  collect  |  send [YYYYMMDD ]
```

The vmstat_collect.sh script is called with either the collect or send argument.

collect : This argument should only be used in the crontab entry that is placed in root's cron file when the vmstat collector is installed.

send : This argument can be used to send the nightly vmstat report file to the SmartCloud Cost Management Server. A log of the FTP transfer is written to /<SCCM_install_dir>/data/YYYYMMDD_ftp.log.

Uninstalling the vmstat collector:

This topic describes how to uninstall the vmstat collector.

About this task

The SmartCloud Cost Management vmstat Collector can be uninstalled by removing the `vmstat_collect.sh` crontab entry from root's crontab. Once that is done, all vmstat collector files and scripts can be removed.

Windows Disk data collector

The Windows disk data collector collects metering data on Windows disk and file systems.

The Windows disk collector collects three types of metering data:

- **File system metrics**: These consist of metrics collected from a specified file system, such as the amount of disk space used by each top level folder with a specified drive or directory, and the number of files (including file in subdirectories) within each of these directories.
- **Physical disk metrics**: These consist of metrics collected from physical disks within a specified Windows machine, such as the disk size. Physical disks are usually divided into one or more logical disks.
- **Logical disk metrics**: These consist of metrics collected from logical disks within a specified Windows machine, such as the disk size and disk space used.

This collector does not require a usage metering file to produce CSR files. The files are produced by the collector's executable program, `WinDisk.exe`. This program is located in `<SCCM_install_dir>\collectors\windisk`. WinDisk runs on the Windows platform only. The .NET Framework Version 3.5 SP1 Redistributable Package is required on each machine where the `WinDisk.exe` collector is run. Windisk can run from either the SmartCloud Cost Management server, where it collects information on remote Windows machines, or directly on a remote machine, and the generated CSR file can be pushed or pulled to the SmartCloud Cost Management processing server.

Identifiers and resources collected by the Windows Disk collector

Identifiers and resources collected by the Windows® Disk collector are converted into a CSR file which can be consumed by SmartCloud Cost Management.

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 standard Rate table.

Table 126. Default Windows Disk Identifiers and Resources (File System Collection)

Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers	
Feed: This is defined by the `Feed` parameter in the Windows Disk collector job file.	-
Folder: This is defined by the `PathToScan` parameter in the Windows Disk collector job file.	-
Resources	

Table 126. Default Windows Disk Identifiers and Resources (File System Collection) (continued)

Identifier Name or Resource Description in SmartCloud Cost Management	**Assigned Rate Code in SmartCloud Cost Management**
MS Windows Disk Folder Usage in GB	DISKSIZE (GB days)
MS Windows Files in Folder	DISKFILE

Table 127. Default Windows Disk Identifiers and Resources (Physical Disk Collection)

Identifier Name or Resource Description in SmartCloud Cost Management	**Assigned Rate Code in SmartCloud Cost Management**
Identifiers	-
Feed: This is defined by the `Feed` parameter in the Windows Disk collector job file.	-
HostName: This is the DNS host name of target machine	-
IPAddress: This is the IP address of target machine	-
DiskName: This is the name by which this disk is known to the system; for example, 'PhysicalDrive0'.	-
DiskDescription: This is a description of the disk; for example 'IDE Hard Drive'	-
DiskIndex: This is the Physical drive number; for example, '1'	-
DiskInterfaceType: This is the Interface type of the drive; for example 'SCSI'.	-
Resources	
MS Windows Physical Disk Size in GB	WDPDSIZE

Table 128. Default Windows Disk Identifiers and Resources (Logical Disk Collection)

Identifier Name or Resource Description in SmartCloud Cost Management	**Assigned Rate Code in SmartCloud Cost Management**
Identifiers	-
Feed: This is defined by the Feed parameter in the Windows Disk collector job file.	-
HostName: This is the DNS host name of target machine	-
IPAddress: This is the IP address of target machine	-
DiskName: This is the label by which the disk is know; for example, 'C:'.	-
DiskDescription: This is a description of the logical disk; for example, 'Local Fixed Disk'	-
DiskFileSystem: This is the type of file system used by the logical disk; for example 'NTFS'	-
DiskVolumeName: This is the volume name of the logical disk; for example, "FINANCIAL_DATA'	-
Resources	
MS Windows Logical Disk Size in GB	WDLDSIZE
MS Windows Logical Disk Used in GB	WDLDUSED

Setting up Windows Disk data collection

This topic provides information about setting up Windows Disk data collection.

SmartCloud Cost Management includes a sample job file, `SampleWinDisk.xml`, that you can modify and use to process the storage data. After you have modified and optionally renamed the `SampleWinDisk.xml` file, move the file to the `<SCCM_install_dir>\jobfiles` directory.

Note that in the `SampleWinDisk.xml` file, the collection steps contain the child elements `WinDisk` and `Parameters`. When Job Runner is run, the `WinDisk` 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 the following table.

Table 129. WinDisk Attributes

Element	Attributes	Description/Values
WinDisk	**fileName**	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. **Note:** If you provide a full path, the path must be an existing path unless you include the attribute `createPath= "true"`.
	ovewrite	Specifies whether the file should overwrite an existing file. Valid values are: • `"true"` (the existing file is overwritten) • `"false"` (the file is not overwritten and the step fails) The default is `"true"`.
	autoRemove (optional)	Specifies whether the file should be automatically removed after the step has executed. Valid values are: • `"true"` (the file is removed) • `"false"` (the file is not removed) The default is `"false"`.
	createPath (optional)	This attribute works in conjunction with the `fileName` attribute. If you include a full path for `fileName`, but the path does not exist, this attribute specifies whether the path is automatically created. Valid values are: • `"true"` (the path is created) • `"false"` (the path is not created) The default is `"false"`.
Collector	**Collector name**	The collector name. *Do not change this parameter*.

Table 129. WinDisk Attributes (continued)

Element	Attributes	Description/Values
	instanceName	The name of the instance for the collector. You can assign any name that has meaning for your organization. For example, the server and drive that you are collecting from.
	instanceDescription	A description of the instance for the collector.
	active	Specifies whether the instance is included in processing. Valid values are: • `"true"` (the instance is processed) • `"false"` (the instance is not processed) The default is `"true"`.
Parameter	**LogDate**	The Windows Disk collector collects data that is current as of the date and time that the collector is run by the SmartCloud Cost Management 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, use `%RNDATE%` as the `LogDate` parameter and include the parameter `LogDate= "RNDATE"` at the job or process level in the job file.
	Retention	This attribute is for future use.
	Feed	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. This is the CSR file that is processed by the Scan program. This parameter is included as an identifier in the CSR file.

Table 129. WinDisk Attributes (continued)

Element	Attributes	Description/Values
	OutputFolder	The process definition folder for the collector. This is the location of the final CSR file that is created by the Scan program.
	LogFileName (optional)	The name of the output log file that will contain CSR records. Usually the log file name is built from the `OutputFolder` and `LogDate` parameters. The `LogFileName` parameter overrides the `OutputFolder` and `LogDate` parameters when generating the name of the log file to which records are written.
	OverwriteLogFile (optional)	When set to 'true', an existing log file will be overwritten. When set to 'false', an existing log file will have new records appended. This parameter is useful when you are collecting Physical and Logical disk and want both metrics to appear within a single file. For example, physical disk collection would first run with `OverwriteLogFile` set to 'true', and then logical disk collection would be run with `OverwriteLogFile` set to 'false'.
	PathToScan (for File System collection only)	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 129. WinDisk Attributes (continued)

Element	Attributes	Description/Values
	Units (optional)	If the attribute is set to GB, is left blank, or is not included, disk space usage is presented in gigabytes. To present the usage units in another measurement, enter one of the following values: • bytes • KB (kilobytes) • MB (megabytes) • A number by which you want to divide the usage units. In this case, the units are measured in bytes rather than gigabytes.
	NumberOfLevels (optional) (for File System collection only)	This attribute works in conjunction with the PathToScan attribute to determine the folder level that will be scanned. For example, if the PathToScan is All (scan all drives) and the NumberOfLevels attribute is 2, the data collection will reflect all second level folders under the scanned drives.
	CollectionType (optional)	Specified the type of metrics to be collected. Defined values are: • FileSystem • PhysicalDisks • LogicalDisks The default is FileSystem.
	ComputerName (optional) (for Physical Disk, Logical Disk collection only)	The name of the computer where physical and logical metrics should be collected from. The ComputerName can also be the IP address of a machine. The default is the local machine.
	UserID (optional) (for Physical Disk, Logical Disk collection only)	The userid needed to access the remote machine. If the UserID parameter is specified, the UserPassword must also be specified otherwise the UserID parameter is ignored. The default is to use the user credentials of the user running WinDisk.
	UserPassword (optional) (for Physical Disk, Logical Disk collection only)	The user password needed to access the remote machine. The password can be clear text or can be encrypted using the SmartCloud Cost Management passwordManager utility. The default is to use the user credentials of the user running WinDisk.

File system collection

Collection of file system metrics is typically done from the SmartCloud Cost Management server by specifying a path to scan, using either a drive letter or a UNC path to a remote Windows system.

A sample job file named `SampleWinDisk.xml` is provided with SmartCloud Cost Management to demonstrate how to use the WinDisk collector to collect file system metrics from Windows file paths.

There are two security considerations when running WinDisk File System collection.

- The user credentials of the user running the WinDisk collector will be used to perform the file system scan. If the user running WinDisk does not have sufficient authority to scan the specified path, WinDisk may be unable to collect metrics, or will collect incomplete metrics.
- WinDisk requests "Backup files and directories" permission when running. This permission allows WinDisk access to the file system in order to collect size metrics. To enable this permission, the user running WinDisk should be added to the "Backup Operators' group. Alternatively, use the **Windows Local Security Policy** tool and in the **User Rights Assignment** section, add the WinDisk user to the "Backup Files and directories" policy.

Physical and logical disk collection

Collection of Physical and Logical disk metrics is done using the Windows Management Instrumentation (WMI) facility within Windows. WMI services must be enabled on the SmartCloud Cost Management server as well as the machine where metrics will be collected from. The WMI Win32_DiskDrive class is used as the source for physical disk metrics, while the WMI Win32_LogicalDisk class is used as the source for logical disk metrics.

The WinDisk collector can be run from either the SmartCloud Cost Management server or directly on the remote Windows machine.

Running from the SmartCloud Cost Management Server

A sample job file named `SampleWinDisk_DiskInfo.xml` is provided with SmartCloud Cost Management to demonstrate how to use the WinDisk collector to collect physical and logical disk metrics from a remote machine. The job file assumes the WinDisk collector is located only on the SmartCloud Cost Management processing server.

The job file has two collection steps: one for physical disk metrics and one for logical disk metrics. The parameters for both steps should be modified for your installation. The parameters which should be changed for your installation are:

- ComputerName
- UserID/UserPassword (if a remote ComputerName is specified)

In the first collection step (Physical Disks) the `OverwriteLogFile` parameter is "true" and the generated log file is overwritten if it exists. In the second collection step (Logical Disks), the `OverwriteLogFile` parameter is "false" and the generated log file is appended to if it exists. If your installation requires separate log files, change the `OverwriteLogFile` parameter to "true" in both collection steps, and specify a complete filename value for the`LogFileName` parameter.

When using Windisk to collect remotely, WMI must be configured properly in order for WinDisk to collect physical and logical disk metrics. This means WMI services must be running on both the SmartCloud Cost Management processing server and the remote computer. In addition, sufficient security rights must be granted so that the user performing the collection on the SmartCloud Cost Management processing server can access the WMI data on the remote machine.

To quickly verify whether the SmartCloud Cost Management processing server can access WMI on the remote machine, run the following command from the command prompt on the SmartCloud Cost Management processing server:

```
wmic /user:username /password:userpassword /node:remoteMachine ComputerSystem get Name
```

Replace the username, userpassword and remoteMachine parameters with the appropriate values for your installation. The remoteMachine parameter is the name of the machine where disk metrics will be collected, and the username/ userpasswords are network credentials used to access the remote computer.

If the WMIC command is run successfully, the name of the remote machine will be returned.

If you receive an error, it likely means the WMI is not configured for communication between the SmartCloud Cost Management processing server and the remote machine. If this is the case, the WinDisk collector will not be able to collect physical and logical disk metrics.

There are many potential reasons why WMI may not be available, including user permissions, firewalls, and/or system configuration. Microsoft provides references and tools to help get WMI configured and running. Please see the following references for additional help.

1. Microsoft WMI Diagnosis Tool:

   http://www.microsoft.com/downloads/details.aspx?familyid=d7ba3cd6-18d1-4d05-b11e-4c64192ae97d&displaylang=en&displaylang=en
2. Microsoft WMI Frequently Asked Questions:

   http://www.microsoft.com/technet/scriptcenter/resources/wmifaq.mspx

Running on a remote Windows machine

Running WinDisk on a remote Windows machines requires the following steps first be taken:

1. If it is not already installed, install the .NET Framework Version 3.5 SP1 Redistributable Package. This package is available electronically from Microsoft, and is also provided with newer Windows versions.
2. Create a directory on the remote Windows machine to hold the WinDisk collector files (for example, C:\WinDisk).
3. Create a directory on the remote Windows machine to hold the WinDisk collector log files (for example, C:\WinDisk\Local).
4. Copy the `WinDisk.exe`, `SampleDiskInfo_LogicalDisks.xml`, `SampleDiskInfo_PhysicalDisks.xml` and `WinDiskMessages.dll` to the `WinDisk` collector directory.
5. Modify the `SampleDiskInfo_LogicalDisks.xml` and `SampleDiskInfo_PhysicalDisks.xml` for parameters needed for your installation. The following parameters should be verified and modified for your installation:
   - `Feed`

- `OutputFolder`
- `ComputerName`

6. Schedule the `WinDisk.exe` binary using Windows Task Scheduler or another scheduling tool so that WinDisk collection takes place on a regular basis. SmartCloud Cost Management usage collection normally takes place once a day.
7. WinDisk requires several parameters to be provided on the command line. The command line usage is:

   ```
   WinDisk XMLFilename CollectorName CollectorInstance
   ```

   For example, using the `SampleDiskInfo_LogicalDisks.xml` job file, the command line is:

   ```
   WinDisk.exe the SampleDiskInfo_LogicalDisks.xml WinDisk Local
   ```

   Similarly, using the `SampleDiskInfo_PhysicalDisks.xml` job file, the command line is:

   ```
   WinDisk.exe the SampleDiskInfo_PhysicalDisks.xml WinDisk Local
   ```

8. Perform the file transfer of the log files to the SmartCloud Cost Management processing server.
9. Clean up the log files. The cleanup step removes old log files.

Windows Process data collector

The Windows Process collector gathers usage data for processes running on Windows 2000/2003/2008 and XP operating systems and produces a log file of the data. This collector runs on the Windows platform only.

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.

In addition, the WPC can be configured to produce a log file that contains system-wide CPU and memory metrics, such as:

- Total system-wide memory, available and used.
- Total system-wide CPU usage and kernel/user/idle time seconds.

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 gathers usage data for processes running on Windows 2000/2003/2008 and XP operating systems and produces a log file of the data.

The Windows Process collector includes a Windows service that supports the collector. The service name is SmartCloud Cost Management 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 must set the policies at both the domain and local level if you are using a domain. Policies for the domain override local policies.

Assigning Polices at the domain level:

This topic describes how to assign security polices at the domain level.

About this task

The following are the steps required to assign security 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:

Procedure

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

This topic describes how to assign security polices at the local level.

About this task

Note: If you install the Windows Process collector on multiple servers, the collector 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.

Procedure

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.

Results

If you are using NTFS, make sure that the account has the permission 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

This topic describes the system configurations that you can use to process the log files produced by 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 SmartCloud Cost Management application 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 SmartCloud Cost Management application server for processing.

Configuration 2: Copying log files to the SmartCloud Cost Management application server using a script

In this configuration, the log files are written to a log directory 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 server. The log files on the central server are then processed into CSR files.

Note: The SmartCloud Cost Management installation does not include a script for transferring files.

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 (for example, *%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.
- If you are using Windows Scripting, use the `WshShell` object to enable the script to get the variable values directly from the environment.

Configuration 3: Writing log files directly to the SmartCloud Cost Management application server

In this configuration, the log files are written directly to a log directory on the SmartCloud Cost Management application 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.

Installing the Windows Process collector

To use the Windows Process collector, you must have the collector installed on the SmartCloud Cost Management application server. In addition, you must install the Windows Process collector on each computer from which you want to collect process data. (In most cases, you will want to collect data for computers other than the application server.)

Note: If you are on an earlier release, for example IBM Tivoli Usage and Accounting Manager 7.3 and you are upgrading to IBM SmartCloud Cost Management 2.1.0.5, then there is no need to upgrade the Windows Process Collector. Nothing has changed in the Windows Process Collector between these releases, except the product name on the installer panels. However, if you have upgraded to the latest release and you do want to upgrade the Windows Process Collector, you must:

1. Uninstall the old version of the Windows Process Collector.
2. Install the Windows Process Collector using one of the methods described below.

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 SmartCloud Cost Management application server does not include SmartCloud Cost Management Processing Engine, which processes the CSR files created by the collector and loads the output data into the database. CSR files must be processed on the application server.

Installing remotely:

This topic provides 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 <*SCCM_install_dir*>\samples\jobfiles\ folder. This file is named SampleDeployProcessCollector.xml and can be modified for your organization.

After you have modified and optionally renamed the SampleDeployProcessCollector.xml file, move the file to the <*SCCM_install_dir*>\jobfiles directory.

The SampleDeployProcessCollector.xml job 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.

SampleDeployProcessCollector.xml job file structure

The following table describes the parameters that are specific to this job file.

Table 130. SampleDeployProcessCollector.xml Job File Parameters

Parameter	Required or Optional	Description
Parameter Action	Required	Do not change this parameter.
Parameter Host	Required	The IP address or DNS name of the server on which you would like to install the Windows Process collector.
Parameter UserId Parameter Password	Required	The Windows user account and password for the host server. The user account must belong to the Administrators group. You can use either a clear text password or an encrypted password in the Parameter Password value. To encrypt the password, use the SmartCloud Cost Management password encryption feature.
Parameter KeyFilename	Optional	If you are using the ssh (Secure Shell) protocol (see the Protocol parameter description), the SSH server's host key.
Manifest	Optional	The path to the DeploymentManifest.xml or DeploymentManifestX64.xml file. This parameter is required only if the file is in a location other than the <*SCCM_install_dir*>\collectors\winprocess directory on the SmartCloud Cost Management application server. The DeploymentManifest.xml or DeploymentManifestX64.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. It is recommended that you do not change the default values in the manifest file.
Protocol	Required	The protocol used to deploy the installation files. This must be set to “win', for Windows. Use of other protocols, such as ssh, is not supported on for Process Collector deployment since protocols other than “win” do not support Windows-specific features, such as registry entries and service configuration. win (Windows)

Table 130. SampleDeployProcessCollector.xml Job File Parameters (continued)

Parameter	Required or Optional	Description
RPDParameters	Optional	These are the same parameters that are in the DeploymentManifest.xml or DeploymentManifestX64.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. **Note:** The DeploymentManifest.xml file is for 32-bit systems and the DeploymentManifestX64.xml file is for 64-bit systems. The following is a brief description of each of these parameters: • InstallPathRemote - The installation path for the collector on the remote system. The default is %WPCInstallPathRemote%tuamprocesscollector\. • 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. • 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 %WPCInstallPathRemote %tuamprocesscollector\logs\. • LogFilePrefix - The default name for the log file is ProcessLog-yyyymmdd.txt. You can use the default prefix "ProcessLog-" 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 delimiter used to separate fields in the log file. The default is 9, a tab (ASCII 9). • WriteProcessRecords-The default “yes” specifies that process records are included in the log file. If you set this parameter to no, process records are not included in the log file. • WriteSystemCPURecords - The default “yes” specifies that System CPU metric records are included in the log file. If you set this parameter to no, System CPU records are not included in the log file. • WriteSystemMemoryRecords - The default “yes” specifies that System Memory records are included in the log file. If you set this parameter to no, System Memory records are not included in the log file.

Table 130. SampleDeployProcessCollector.xml Job File Parameters (continued)

Parameter	Required or Optional	Description
`Verbose`	Optional	The value `"true"` specifies that additional information is included in the job log file for debugging and troubleshooting purposes. If you set this parameter to `"false"`, do not include this parameter, or leave the parameter value blank, this additional information is not provided in the log file.
`SourcePath`	Required	The path to the *<SCCM_install_dir>*`\collectors\winprocess` directory. This directory contains the installation file. Do not change this value unless the `winprocess` folder is not in the default location.
`UseSFTP`	Optional	If the `Protocol` parameter is set to `"ssh"`, use this parameter to specify whether the SFTP or SCP protocol will be used to transfer files. Valid values are: • `"true"` (the SFTP protocol is used) • `"false"` (the SCP protocol is used, this is the default) If you do not include this parameter or leave the parameter value blank, the SCP protocol is used. A value of `"true"` is used with certain SSH servers (such as Tectia SSH Server 5.x) to allow file transfers to complete successfully. This parameter is not shown in the `SampleDeployProcessCollector.xml` job file.

Running the deployment job file

To run the deployment job file from the command prompt, use the following command:

`startJobRunner.bat` *<deployment file name>*`.xml`

Where *<deployment file name>* is the name that you give to the deployment job file.

To run the deployment job file from Administration Console:

1. In Administration Console, click **Task Management** > **Job Runner** > **Job Files**.
2. On the Job Runner Job Maintenance page, click the **View popup menu** icon for the job file that you want to run and then click **Run Job**.

Note: Make sure that the deployment file is in the *<SCCM_install_dir>*`\jobfiles` directory so that it will be displayed on the Job Runner Job Maintenance page.

Installing manually:

To install the Windows Process collector using the GUI, complete the following steps in the installation wizard.

Procedure

1. Log on to the Windows operating system as a user in the Administrator group.
2. Click the Windows **Start** button and then click **Run**.
3. Type the path to the setup program for the collector package and then click **OK**. The installer executable is located in `<SCCM_install_dir>/collectors/winprocess/setup-SCCM-wpc-2-1-0-1-windows_32_64.exe`. The 32-bit and 64-bit (x64 and Itanium) versions of the Windows Process Collector are in the single

installation binary. The installer automatically determines the type of processor in use during installation. The installation wizard opens.

4. On the welcome page, click **Next**.
5. On the license agreement page, click **I accept both the IBM and the non-IBM terms** and then click **Next**. You must accept the terms to continue the installation.
6. On the installation location page, choose the default location for installation; type a location; or click **Browse** to choose a location. Click **Next** after making your selection. Note that the location path can contain ASCII characters only. DBCS (double-byte character set) characters are not supported.
7. On the **Select the record types to be written** page, complete the following and then click **Next**.
   a. Select the type of records the Windows Process Collector should write to its log file using the check boxes for Process, System CPU and System Memory. Records will be written to the log file if a check box is selected for a particular record type. By default, the Process record type is selected. You must select at least one record type.
8. On the log parameters page, select from the following settings and then click **Next**:
   - **Start application after install and during reboot**: The Windows Process collector includes a Windows service that supports the collector. If this check box is selected (the default), the service will start automatically after the collector is installed or the server is restarted. If you do not want the service to start automatically, clear this check box.
   - **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 type 5 and select **seconds** in the drop-down list, the collector checks every 5 seconds to determine which new processes have begun since the last check and tracks those processes until completion.
   - **Use local time in logging 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 that is produced by the collector. 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 this check box is selected.
   - **Log file prefix**: The default name for the log file produced by the collector is `ProcessLog-yyyymmdd.txt`. You can use the default prefix `ProcessLog-` or replace it with the prefix of your choice (or no prefix).
   - **Log file path**: Type the path to the folder that you want to store the process log in or click **Browse** to choose a location.
9. On the interval accounting parameters page, select from the following settings and then click **Next**:
   - **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 only a cumulative End record. This is especially beneficial for long-running processes that begin in one billing period and end in another billing period. 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 or Write accounting records at boxes 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 `winprocess.wsf` script. For more information, contact IBM Software Support.
- **Run this command before each interval**: Type a command (for example, a command to run a .bat file or an executable) that will be executed at each logging interval.
- **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 24 hours, an interval record will be created every 24 hours for each process that is running until the process ends. If you want to create interval records for all processes running at a specified time of day, use the **Write accounting records at** option.
- **Write accounting records at**: This option enables you to set a time each day (in 24-hour format) to produce interval records. An interval record is created for each process running at this time. This option is intended to be used to track longer running processes such as SQL Server, IIS, and other services. For these types of processes, you might want to create one daily interval record.

10. On the summary information page, review the installation information and then click **Install** if you want to continue the installation. A series of progress messages are displayed as the installation continues.
11. On the successful installation page, click **Finish**.

Windows Process collector log file format

This topic 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.

  If the process ends before the interval accounting begins, an interval record is created showing the time that the interval ran. Likewise, if the process ends between intervals, a final interval record is created showing the time that the interval ran.
- End records, which provide summary usage data at the end of a process. All totals in an End record are cumulative for the whole process.

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

End records appear in the following situations:

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

Note: The term "process" in the following table can refer to the entire process, or the start, interval, or end of a process depending on whether interval accounting is used.

Table 131. Windows Process Collector Log File Format - Process Records

Field Name	Description/Values
Record Type	S = Start of process (note that this does not appear if interval accounting is not used). I = Interval (note that this does not appear if interval accounting is not used). E = End of process (this record appears if you do not enable interval accounting or if you enable interval accounting and write end records).
ProcessID	Process identifier (PID) assigned to the process by the operating system.
ParentProcessID	The PID for the entity that created the process. Assigned by the operating system.
ProcessName	The name of the process.
ProcessPath	The path where the process executable is located.
MachineName	The name of the computer running the process.
UserName	The name of the user that created the process.
TerminalServicesSessionID	If Microsoft Terminal Services is used to access the process on the computer, the session ID.
CreateDateTime	The date and time that the process was created.
ExitDateTime	The date and time that the entire process ended.
ExitCode	The result from the completion of the process.
IntervalStartDateTime	If using interval accounting, the date and time that the interval started.
IntervalEndDateTime	If using interval accounting, the date and time that the interval ended.
ElapsedTimeSecs	The total elapsed time in seconds for the process.
CPUTimeSecs	The total elapsed CPU time in seconds for the process. This field is the sum of KernelCPUTimeSecs and the UserCPUTimeSecs fields.
KernelCPUTimeSecs	The total elapsed time in seconds that the process spent in kernel mode.
UserCPUTimeSecs	The total elapsed time in seconds that the process spent in user mode.
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.

Table 131. Windows Process Collector Log File Format - Process Records (continued)

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

Table 132. Windows Process Collector Log File Format – System CPU Records

Field Name	Description/Values
Record Type	SYSCPUINFO_START = Start of System CPU usage logging (note that this does not appear if interval accounting is not used). SYSCPUINFO_INTERVAL = Interval record of System CPU usage (note that this does not appear if interval accounting is not used). SYSCPUINFO_END = End of System CPU usage recording (this record appears if you do not enable interval accounting or if you enable interval accounting and write end records).
MachineName	The name of the computer where the collector is running.
HostName	The host name of the computer where the collector is running.
IPAddress	The IPv4 or IPv6 address where the collector is running.
IntervalStartDateTime	The date and time of the start of the interval where CPU usage is measured.

Table 132. Windows Process Collector Log File Format – System CPU Records (continued)

Field Name	Description/Values
IntervalEndDateTime	The date and time of the end of the interval where CPU usage is measured.
TotalCPUAVGUsage	The average percentage of system-wide CPU usage during the interval.
TotalCPUMINUsage	The minimum percentage of system-wide CPU usage during the interval.
TotalCPUMAXUsage	The maximum percentage of system-wide CPU usage during the interval.
UserTimeSecs	The total elapsed time in seconds that all processes spent in kernel mode during the interval.
UserTimeSecs	The total elapsed time in seconds that all processes spent in user mode during the interval.
NumberOfProcessors	The number of processors on the computer.

Table 133. Windows Process Collector Log File Format – System Memory Records

Field Name	Description/Values
Record Type	SYSMEMINFO_START = Start of System Memory usage logging (note that this does not appear if interval accounting is not used). SYSMEMINFO_INTERVAL = Interval record of System Memory usage (note that this does not appear if interval accounting is not used). SYSMEMINFO_END = End of System Memory usage recording (this record appears if you do not enable interval accounting or if you enable interval accounting and write end records).
MachineName	The name of the computer where the collector is running.
HostName	The host name of the computer where the collector is running.
IPAddress	The IPv4 or IPv6 address where the collector is running.
IntervalStartDateTime	The date and time of the start of the interval where CPU usage is measured.
IntervalEndDateTime	The date and time of the end of the interval where CPU usage is measured.
PhysicalMemoryTotal	The total amount of physical memory available in the computer in megabytes.
AvailableMemoryAVG	The average amount of system-wide memory available during the interval in megabytes.
AvailableMemoryMIN	The minimum amount of system-wide memory available during the interval in megabytes.

Table 133. Windows Process Collector Log File Format – System Memory Records (continued)

Field Name	Description/Values
AvailableMemoryMAX	The maximum amount of system-wide memory available during the interval in megabytes.
UsedMemoryAVG	The average amount of system-wide memory used during the interval in megabytes.
UsedMemoryMIN	The minimum amount of system-wide memory used during the interval in megabytes.
UsedMemoryMAX	The maximum amount of system-wide memory used during the interval in megabytes.
UsedMemoryPctAVG	The average amount of system-wide memory used during the interval in percentage terms.
UsedMemoryPctMIN	The minimum amount of system-wide memory used during the interval in percentage terms.
UsedMemoryPctMAX	The maximum amount of system-wide memory used during the interval in percentage terms.

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), and so on.

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

Identifiers and resources defined by the Windows Process collector

The Windows Process collector gathers usage data for processes running on Windows 2000/2003/2008 and XP operating systems.

By default, the following fields in the Windows Process collector log file are defined as chargeback identifiers and resources). The rate codes assigned to the resources are pre-loaded in the Rate table.

Table 134. Default Windows Process Identifiers and Resources - Process

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers		
-	Feed (defined in the Windows Process collector job file)	-
ProcessName	ProcessName	-
ProcessPath	ProcessPath	-
MachineName	Server	-

Table 134. Default Windows Process Identifiers and Resources - Process (continued)

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
UserName	UserName	-
PriorityClass	PriorityClass	-
BasePriority	BasePriority	-
Resources		
ElapsedTimeSecs	MS Windows Elapsed Time	WINELPTM
CPUTimeSecs	MS Windows CPU Time	WINCPUTM
KernelCPUTimeSecs	MS Windows Kernel CPU Time	WINKCPUT
UserCPUTimeSecs	MS Windows User CPU Time	WINCPUUS
Read Requests	MS Windows Read Requests	WINRDREQ
KBytesRead	MS Windows KB Read	WINKBYTR
Write Requests	MS Windows Write Requests	WINWRREQ
KBytesWritten	MS Windows KB Written	WINKBWRI
PageFaultCount	MS Windows Page Fault Count	WINPGFLT

Table 135. Default Windows Process Identifiers and Resources - System CPU

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers		
	Feed (defined in the Windows Process collector job file)	
MachineName	MachineName	
HostName	HostName	
IPAddress	IPAddress	
Resources		
TotalCPUAVGUsage	Total System-Wide CPU Utilization (%) - average	WPCCTAVG
KernelTimeSecs	Total System-Wide Kernel Time	WPCKRNLT
UserTimeSecs	Total System-Wide User Time	WPCUSERT
UserTimeSecs	Total System-Wide Idle Time	WPCIDLET
UserTimeSecs	The number of processors on the computer.	WPCCPUS

It is recommended that this data is written to the resource utilization table. For more information on how to do this, see *DBLoad* topic in the *Reference guide*.

Table 136. Default Windows Process Identifiers and Resources - System Memory

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Identifiers		
	Feed (defined in the Windows Process collector job file)	
MachineName	MachineName	
HostName	HostName	
IPAddress	IPAddress	

Table 136. Default Windows Process Identifiers and Resources - System Memory (continued)

Log File Field	Identifier Name or Resource Description in SmartCloud Cost Management	Assigned Rate Code in SmartCloud Cost Management
Resources		
PhysicalMemoryTotal	The total amount of physical memory available in the computer in megabytes.	WPCMTOT
AvailableMemoryAVG	The average amount of system-wide memory available during the interval in megabytes	WPCMAVL
UsedMemoryAVG	The average amount of system-wide memory used during the interval in megabytes	WPCMUSE
UsedMemoryPctAVG	The average amount of system-wide memory used during the interval in percentage terms.	WPCMUSEP

It is recommended that this data is written to the resource utilization table. For more information on how to do this, see *DBLoad* topic in the *Reference guide*.

Setting up Windows Process data collection

This topic provides information about setting up Windows Process data collection.

SmartCloud Cost Management includes one sample job file, SampleWinProcessIntegrator.xml, that you can modify and use to process the log file that is produced by the Windows Process collector.

SampleWinProcessIntegrator.xml is a sample script that takes advantage of Integrator to read and process Windows Process Collector log files from SmartCloud Cost Management 7.1.3 and later. SampleWinProcessIntegrator.xml can be run on any platform where SmartCloud Cost Management processing is running.

After you have modified and optionally renamed the SampleWinProcessIntegrator.xml file, move the file to the <*SCCM_install_dir*>\jobfiles directory.

Configuring the SampleWinProcessIntegrator.xml job file

This section describes the job file for the Windows Process collector log file processing using the SmartCloud Cost Management Integrator.

The following sample shows the structure of the Integrator step of the job file:

Note: In the following piece of code, a number of code lines are too long to be displayed as unbroken lines. A break in a line is indicated by a backslash '\'.

```
<Step id="Server1 Collection Process Information"
 description="Server1 WinProcess - Process"
 type="ConvertToCSR"
 programName="integrator"
 programType="java"
 active="true">
 <Integrator>
 <Input name="CollectorInput" active="true">
   <Collector name="WINPROCESS">
    <Template filePath="%HomePath%/collectors/winprocess/ \
    StandardTemplates.xml" name="WPC_PROC" />
```

```
  </Collector>
 <Parameters>
  <Parameter name="LogDate" value="PREMON" DataType="String"/>
  <Parameter name="Feed" value="server1" DataType="String"/>
 </Parameters>
 <Files>
  <File name="%HomePath%/samples/logs/collectors/winprocess \
  /ProcessLog-%LogDate_End%.txt" type="input"/>
  <File name="%ProcessFolder%/exception.txt" type="exception" />
 </Files>
 </Input>
 </Integrator>
</Step>
```

The Step element has the following attributes:

Table 137. Step attributes

Attribute	Required or optional	Description/Parameters
`id="step_id"`	Required.	step_id refers to the step ID.
`type="ConvertToCSR"`	Required.	Do not change this value.
`programType="java"`	Required.	Do not change this value.
`programName="integrator"`	Required.	Do not change this value.

The Step element contains the Integrator element, which has no attributes.

The Integrator element contains the Input element, which has the following attributes:

Table 138. Input element attributes

Attribute	Required or optional	Description/Parameters
`name="CollectorInput"`	Required.	Do not change this value.
`active="true" | "false"`	Required.	Set this attribute to "true" to run the collector or "false" if you do not want to run the collector.

The Input element can contain the Collector, Parameters, and Files elements.

The Collector element is required. It has one attribute:

Table 139. Collector element attributes

Attribute	Required or optional	Description/Parameters
`name="WINPROCESS"`	Required.	Do not change this value.

The Collector element contains the Template element.

The Template element is required and has the following attributes:

Table 140. Template element attributes

Attribute	Required or optional	Description/Parameters
`filePath="template_file_path"`	Optional.	The path of the template file. If you do not specify a value, the default `%HomePath%/collectors/winprocess/StandardTemplates.xml` is used. You can generate custom template files and use them in place of the standard templates by specifying a different file name.
`name="template_name"`	Required.	The name of the template. This value controls the type of log records which are read and processed by the job. Valid values are `WPC_SCPU` for System CPU, `WPC_SMEM` for System Memory, and `WPC_PROC` for Process.

The Parameters element has no attributes. The Parameters element must contain the Feed and LogDate parameters.

The Feed parameter contains the following attributes:

Table 141. Feed parameter attributes

Attribute	Required or optional	Description/Parameters
`name="Feed"`	Required.	The name of the parameter.
`value="server1"`	Required.	Specifies the subdirectory of the process folder where CSR or CSR+ files from the same feed are stored.
`dataType="STRING"`	Required.	The data type.

The LogDate parameter contains the following attributes:

Table 142. LogDate parameter attributes

Attribute	Required or optional	Description/Parameters
`name="LogDate"`	Required.	The name of the parameter.
`value="%LogDate_End%"`	Required.	The date for which you are processing usage logs. The value can be a date literal or a date keyword.
`dataType="DATETIME"`	Required.	The data type expected for this parameter.
`format="yyyyMMdd"`	Required.	The simple date time format that is used in the date time value.

The Files element is optional. It has no attributes. The Files element may contain the File parameter.

The File parameter has the following attributes:

Table 143. File parameter attributes

Attribute	Required or optional	Description/Parameters
`name="input_file_path"`	Required.	The name of an input file to be processed. The input file name can include pre-defined macros. For example, `"ProcessLog-%LogDate_End%.txt"`
`type="input"`	Required.	Indicates the file is an input file to be processed.
`name="exception_file_path"`	Required.	The name of a file in which to log exceptions.
`type="exception"`	Required.	The type of data written to the file.

Troubleshooting

Verify that you have met all system specifications for setting up and running the Windows Processor collector. To resolve potential errors that arise with the Windows Processor collector, refer to the troubleshooting information in this topic.

This application has failed to start because MSVCR71.dll was not found

The message `This application has failed to start because MSVCR71.dll was not found` can occur if the Windows system file `MSVCR71.dll` is not in the Windows `system32` directory on the server where the Windows Processor collector is running or in the same directory as the `WinPService.exe` executable.

This message can be reproduced by opening a command prompt, changing the current path to the path where the `WinPService.exe` executable is located, and issuing the command:

```
winpservice.exe -install
```

If the `MSVCR71.dll` file is installed in either the Windows `system32` directory or the same directory as the `WinPService.exe` executable, you will get the message:

```
AUCWP0414I SmartCloud Cost Management Process Collector is installed
```

If the `MSVCR71.dll` file is not found in either the `system32` directory or the or the same directory as the `WinPService.exe` executable, you will get the system message:

```
This application has failed to start because MSVCR71.dll was not found.
```

The problem can be resolved by finding another copy of `MSVCR71.dll` on the system and copying it either to the Windows `system32` directory or to the same directory as the `WinPService.exe` executable.

z/VM data collector

The z/VM® data collector runs on a z/VM system and produces CSR files that can be processed by SmartCloud Cost Management. The CSR files are sent via FTP from the system running the z/VM data collector to the SmartCloud Cost Management application server.

To enable the z/VM collector to run, you must do the following:

- Install and run the data collector on a z/VM system
- Transfer the output CSR file to the SmartCloud Cost Management application server.
- Create a job file on the SmartCloud Cost Management application server to process the CSR file and load it the data into the database
- Run Job Runner.

To set up data collection for Advanced Accounting on an AIX system and to send the files to a Windows system, use the scripts and files provided for the AIX Advanced Accounting Data Collector for Linux and UNIX.

z/VM standard billable items collected

This section describes the z/VM resources that are collected for chargeback.

Table 144. z/VM resources that are collected for chargeback

z/VM resources
• Connect time
• CPU time
• Virtual SIOs
• Virtual cards read
• Virtual lines printed
• Virtual cards punched
• Temporary disk space

z/VM Accounting Records Processed

This topic describes the accounting records that are processed by the z/VM collector.

- Virtual Machine Resource Usage
- Temporary Disk Space Usage

Virtual Machine Resource Usage

RECORD POSITION
CONTENTS

1-8 USERID

9-16 ACCOUNT NUMBER

17-28 DATE AND TIME (MmddYYhhMMSS)

29-32 SECONDS OF CONNECT TIME

33-36 MILLISECONDS OF PROCESSING TIME*

37-40 MILLISECONDS OF VIRTUAL PROCESSOR TIME

41-44 NUMBER OF PAGE READS

45-48 NUMBER OF PAGE WRITES

49-52 NUMBER OF VIRTUAL MACHINE SIOs FOR NON-SPOOLED I/O

53-56 NUMBER OF CARDS SPOOLED TO PUNCH

57-60 NUMBER OF LINES SPOOLED TO PRINTER

61-64 NUMBER OF CARDS SPOOLED FROM READER

65-78 RESERVED

79-80 CARD IDENTIFICATION = 01

* This field includes the time for VM supervisor functions.

The data in record positions 1-28 and 79-80 is character, all other fields are hexadecimal.

Temporary Disk Space Usage

The following lists the z/VM accounting records that the z/VM collector processes:

RECORD POSITION
CONTENTS

1-32 SAME AS RESOURCE USAGE RECORD

33 DEVICE CLASS

34 DEVICE TYPE

35 MODEL (IF ANY)

36 FEATURE (IF ANY)

37-38 NUMBER OF TEMPORARY DISK CYLINDERS*

37-40 NUMBER OF TEMPORARY DISK BLOCKS (FBA)*

39-78 UNUSED

79-80 CARD IDENTIFICATION = 03

* If DEVICE CLASS = FBA X'01', then 37-40 contains number of FBA blocks.

The data in record positions 1-28 and 79-80 is character, all other fields are hexadecimal.

Creating a process definition directory

Before you attempt to transfer CSR files to the SmartCloud Cost Management application server, make sure that you have created a process definition directory and the applicable number of feed subdirectories. Process definition directories contain the files required to process usage data from a particular source such as a database, operating system, or application. The CSR files that are created by the z/VM data collector are sent via FTP from the system running the z/VM data collector to a process definition folder on the SmartCloud Cost Management application server system.

CSR files must be sent to a feed subdirectory within the process definition directory. The feed subdirectory might represent the z/VM system from which the CSR files are sent. For example, if the system VM01 is sending CSR files, you might create a process definition directory named `zVM` that contains the feed

subdirectory VM01. The CSR files are named yyyymmdd.txt.

Installing the z/VM collector

This section describes how to install and run the data collector on a z/VM system.

The z/VM data collector is distributed using the program module CIMSCMS. The CIMSCMS program is written in COBOL and ASSEMBLER and is run on z/VM.

To install the z/VM collector, install the following files on a z/VM disk. These files are in the <*SCCM_install_dir*>\collectors\zvm folder.

Note: If you transfer the installation files to the z/VM system via FTP, the target file must be fixed block with a record size of 80.

FILE NAME	TYPE	DESCRIPTION
CIMSVMT	TXTLIB	Contains all the object modules needed to build the CIMSCMS executable program.
CIMSVME0	REXX	Execute for CIMSCMS.
CIMSVMDA	Data	Control statements first CIMSCMS
CIMSVMDB	Data	Control statements second CIMSCMS
CIMSVMD5	Data	Calendar

Binding the CIMSCMS program:

This section describes the process for binding the CIMSCMS Program for the z/VM data collection process.

The CIMSCMS program was compiled and assembled in a z/VM environment. The Language Environment® (LE) facility of z/VM is required for CIMSCMS.

To define the CIMSVMT TXTLIB and LE library into the global TXTLIB, execute the following command:

```
GLOBAL TXTLIB CIMSVMT SCEELKED
```

Then bind the CIMSCMS module using the following command:

```
BIND CIMSCMS (MAP
```

The preceding command builds the CIMSCMS module on the z/VM disk.

Running the z/VM Collector

This topic describes how to run the CIMSCMS program.

The following file is used to execute REXX code that runs CIMSCMS:

```
/* REXX */
'vmfclear'
'desbuf'
/*
                    ____________________________________
                   CIMSCMS EXECUTION CONTROL STATEMENTS
                    ____________________________________

  THIS EXEC READS THE VM DISK ACCOUNT DATASET & CREATES A CSR FILE.
  The CSR file is processed by TUAM and/or CIMS Server.
```

```
     DISKACNT DATA should be changed to the disk accounting data
'FILEDEF CMSIN    DISK DISKACNT DATA    A1 (RECFM F  LRECL 80'
/*       CMSIN is the z/VM Accounting records                          */

'FILEDEF CIMSPRNT DISK CIMSCMSA LISTING A1 (RECFM FB LRECL 133 BLOCK 133'
'FILEDEF CIMSMSG DISK CMSMSGA LISTING A1 (RECFM F LRECL 129 BLOCK 129'
'FILEDEF CIMSCLDR DISK CIMSVMD5 DATA    A1 (LRECL 80  RECFM F'
'FILEDEF CIMSCNTL DISK CIMSVMDA DATA    A1 (RECFM F  LRECL 80'
/*
   CIMSCMS1 DATA A1 = control record input for the 1st execution.
                                                                 */
'FILEDEF SORTFILE DISK SORTOUT  DATA   A1 (RECFM FB LRECL 80  BLOCK 80'
'CIMSCMS'

if rc -= 0 then do
  src = rc
  if src = 20 then say 'CIMSCMS processing terminated - no records selected'
  else do
     say 'CIMSCMS first pass terminated with error.  Do you wish to see the messages
(y/n)?'
    pull yn
    upper yn
    if left(yn,1) = 'Y' then 'xedit cimscmsa listing'
    end
  exit src
  end
queue '9 16  78 78  1 8  80 80  17 28'
if rc ^= 0 then exit rc
'SORT SORTOUT DATA A1 SORTED DATA A1'
if rc ^= 0 then do
  src = rc
  'desbuf'
  exit src
  end
'FILEDEF CIMSPRNT DISK CIMSCMSB LISTING A1 (RECFM F  LRECL 133 BLOCK 133'
'FILEDEF TUAMCSR DISK TUAMCSR DATA A1 (RECFM V LRECL 516 BLOCK 516'
'FILEDEF CIMSMSG DISK CIMSMSGB LISTING A1 (RECFM F  LRECL 129 BLOCK 129'
'FILEDEF CIMSCNTL DISK CIMSVMDB DATA    A1 (RECFM F  LRECL 80'
/*
   CIMSVMDB DATA A1 IS THE CONTROL RECORD INPUT FOR SECOND EXECUTION.
                                                                 */
'CIMSCMS'
if rc -= 0 then do
   src = rc
   if src = 20 then say 'CIMSCMS processing terminated - no records selected'
   else do
     say 'CIMSCMS 2nd pass terminated with error.  Do you wish to see',
       'the messages (y/n)?'
     pull yn
     upper yn
     if left (yn, 1) = 'Y' then 'xedit cimscmsgb listing'
     end
   end
'ERASE SORTED   DATA  A1'
'ERASE SORTOUT DATA  A1'
say 'Do you wish to view the report file from the first run (y/n)?'
parse pull yn
upper yn
if left(yn,1) = 'Y' then 'XEDIT CIMSCMSA LISTING A1'
say 'Do you wish to view the report file from the second run (y/n)?'
parse pull yn
upper yn
if left(yn,1) = 'Y' then 'XEDIT CIMSCMSB LISTING A1'
say 'Do you wish to view the created CSR records (y/n)?'
parse pull yn
upper yn
```

```
if left(yn,1) = 'Y' then 'XEDIT TUAMCSR DATA A1'
/*
  FILE TUAMCSR DATA is input to TUAM and/or CIMS Server                */
```

CIMSCMS Control Statements

The following table displays the CIMSCMS control statements that are required by the z/VM data collector when executing CIMSCMS using CIMSVMEO.

Each control statement starts in column 1 and control statements are separated by a space. Statements that start with a space or asterisk are comments.

Control Statements	Description
ACCOUNT TAG	CSR account number identifier tag name.
ASSIGN ALL RECORDS PRIME	Use prime rate codes, default is non-prime rate codes.
DAILY TRANSACTIONS	CSR records are built for each day change.
DATE SELECTION	Select data by date range.
EXCLUDE	Exclude account codes and user IDs.
EXECUTE	Determine the execution of CIMSCMS, first or second pass.
HDx	Report heading titles.
SELECT	Select account codes and user IDs.
USERID TAG	CSR user ID identifier tag name.

In addition to the preceding control statements, a statement (oldratecode = newratecode) is available to replace standard rate codes created by the z/VM data collector. For more information about this control statement, see Changing Rate Codes Web site..

ACCOUNT TAG name

This control statement assigns the identifier name for the account number in the CSR record. The default value is `Account_Number`.

ASSIGN ALL RECORDS PRIME

This control statement assigns all jobs to the prime shift. No slicing of the execution times over the prime and non-prime shifts will be performed.

DAILY TRANSACTIONS

The control statement `DAILY TRANSACTIONS` specifies that summary billing transactions are to be generated for each change in `DATE`.

When this statement is not present, billing transactions are created when either the account code or the user ID value changes.

DATE SELECTION YYYYMMDD YYYYMMDD

The control statement specifies the `LOW` (from) and `HIGH` (to) selection date for VM/CMS session accounting records. Each session accounting record is compared

with the specified dates. Records that are equal to or greater than the LOW value and equal to or less than the HIGH value are selected for processing.

Example

```
DATE SELECTION 20070101 20070115
```

This statement specifies the selection of records from January 1, 2007 through January 15, 2007.

A CIMS keyword can be placed into FIELD 1.

Control statement keywords automatically calculate specific dates.

The following key words are supported:

1. CURRENT: Sets date range based on current period from CIMS calendar file.
2. PREVIOUS: Sets date range based on previous period from CIMS calendar file.
3. **CURDAY: Sets date range based on run date and run date less one day.
4. **CURWEK: Sets date range based on run week (Sun - Sat).
5. **CURMON: Sets date range based on run month.
6. **PREDAY: Sets date range based on run date, less one day.
7. **PREWEK: Sets date range based on previous week (Sun - Sat).
8. **PREMON: Sets date range based on previous month.

Note: Run date is used to determine current and previous date values.

EXCLUDE ACCOUNT low high and EXCLUDE USERID low high

These control statements are used to exclude account and user IDs from processing. Values inside the range are not processed. A maximum of 200 of each type of EXCLUDE control statements is supported.

Examples

```
EXCLUDE ACCOUNT AABBBB AABBBB
EXCLUDE USERID ABCD
```

EXECUTE

This control statement determines the execution of CIMSCMS (first or second pass). This control statement is required. There are two examples for this control statement.

First pass example:

EXECUTE CIMS VM/CMS 01

Second pass example:

EXECUTE CIMS VM/CMS 02

POSITION	VALUE	DESCRIPTION
1-7	EXECUTE	Control Statement Identifier
8	b	

POSITION	VALUE	DESCRIPTION
9-19	CIMSbVM/CMS	REQUIRED VALUE
20	b	
21-22	XX	The value 01 specifies that raw VM/CMS accounting data is being input and that the data is to be validated and written on the file SORTFILE. The value 02 specifies that sorted VM/CMS accounting data is being input and that the data is to be read from the file SORTFILE. The EXE control statement is required.

b = Blank

HD1, HD2, HD3-Optional Input

Program CIMSCMS prints three lines of headline information each time a new page of printed output is started. These three lines of heading information can be replaced by supplying a control statement in the input stream with HD1, HD2, and/or HD3 in columns 1-3. The information contained in positions 4-72 of each record replaces line 1, line 2, and/or line 3 on the printed output. These records should be the first three control statements in the input stream.

POSITION	VALUE	DESCRIPTION
1-3	HD1, HD2, HD3	Control Statement Identifier
4-72	X(69)	Text

Example

```
HD1                      CIMS, The Enterprise Chargeback System
HD2                        Session Accounting for VM/CMS
HD3                               CIMSCMS 1st pass
```

SELECT ACCOUNT low high and SELECT USERID low high

These control statements are used to select account codes and user IDs for processing. Values outside the range are not processed. A maximum of 200 of each type of SELECT control statement is supported.

Examples

```
SELECT ACCOUNT AABBBB AABBBB
SELECT USERID ABCD
```

USERID TAG name

This control statement assigns the identifier name for the user ID in the CSR record. The default value is User_ID.

About the Z/VM collector output CSR file

This section describes the CSR records that are generated by z/VM for loading into the SmartCloud Cost Management.

Each account record generated by z/VM contains a user ID and account number in two eight-character fields. The data contained in these fields appears in the account control fields used by SmartCloud Cost Management. The CSR records are generated for the unique combinations of these control fields.

By default, the following are defined as chargeback identifiers and resource rate codes in the CSR file. The rate codes assigned to the resources *are not* pre-loaded in the Rate table and must be added to the table using Administration Console.

Identifiers

- User_ID
- Account_Number

Resource Rate Codes

Resource rate codes are specified as prime or non-prime to designate prime or non-prime billing rates. By default, the CSR records contain non-prime rates codes. To use prime rate codes, use the control statement `ASSIGN ALL RECORDS PRIME` .

PRIME CODES	NON-PRIME CODES	DESCRIPTION
ZCM1	ZCV1	CONNECT TIME (SECONDS)
ZCM2	ZCV2	CPU TIME (SECONDS)
ZCM3	ZCV3	VIRTUAL SIO'S
ZCM4	ZCV4	VIRTUAL CARDS READ
ZCM5	ZCV5	VIRTUAL LINES PRINTED
ZCM6	ZCV6	VIRTUAL CARDS PUNCHED
ZCM8	ZCV8	TEMPORARY DISK SPACE - CYL
ZCM9	ZCV9	TEMPORARY DISK SPACE - FBA

Changing Rate Codes

You can redefine the standard rate codes (for example, to create rate codes by system) by supplying the following information in the input control statement data set:

```
oldratecode = newratecode
```

Where `oldratecode` is any one of the VM/CMS rate codes and `newratecode` is the replacement.

Example

In this example, all the standard VM/CMS rate codes will be replaced with rate codes that start with the system code VM01:

```
ZCM1 = VM01ZCM1 CONNECT TIME
ZCM2 = VM01ZCM2 CPU TIME
ZCM3 = VM01ZCM3 VIRTUAL SIO's
ZCM4 = VM01ZCM4 VIRTUAL CARDS READ
```

```
ZCM5 = VM01ZCM5 VIRTUAL LINES PRINTED
ZCM6 = VM01ZCM6 VIRTUAL CARDS PUNCHED
ZCM8 = VM01ZCM8 TEMPORARY DISK SPACE - CYL
ZCM9 = VM01ZCM9 TEMPORARY DISK SPACE - FBA
ZCMX = VM01ZCMV MONEY
ZCV1 = VM01ZCV1 CONNECT TIME
ZCV2 = VM01ZCV2 CPU TIME
ZCV3 = VM01ZCV3 VIRTUAL SIO's
ZCV4 = VM01ZCV4 VIRTUAL CARDS READ
ZCV5 = VM01ZCV5 VIRTUAL LINES PRINTED
ZCV6 = VM01ZCV6 VIRTUAL CARDS PUNCHED
ZCV7 = VM01ZCV7 TEMPORARY DISK SPACE - Money
ZCV8 = VM01ZCV8 TEMPORARY DISK SPACE - CYL
ZCV9 = VM01ZCV9 TEMPORARY DISK SPACE - FBA
```

Transferring output CSR files from the z/VM system

This section describes the process for transferring the output CSR file to the SmartCloud Cost Management application server.

The CSR file must be transferred from the z/VM system to the SmartCloud Cost Management application server using FTP. Establish an SmartCloud Cost Management FTP site to receive host-based CSR data (usually, the FTP root is the <*SCCM_install_dir*>\processes folder).

The CSR file must be placed in the appropriate process definition directory and feed subdirectory on the SmartCloud Cost Management application server system.

Setting up z/VM data collection

This topic provides information about setting up z/VM data collection.

On the SmartCloud Cost Management application server, create an XML job file to collect the CSR file created by the z/VM collector. The job file must contain a `Process` element that specifies the process definition directory that contains the CSR files. Note that the collection step is not required because the process definition directory contains created CSR files. The first step in the job file should call the Scan program.

Setting up for data collection on a Linux or UNIX system

After you have installed the operating system, file system, and Oracle or DB2 database collectors on a UNIX or Linux system, you must set up the system as described in this section.

Note: The Oracle and DB2 database data collectors are provided in IBM SmartCloud Cost Management Enterprise.

Installing SmartCloud Cost Management Data Collectors for UNIX and Linux

To quickly deploy the operating system, file system, and database collector packages to multiple computers, you can install the collectors remotely.

Remote installation performs the following tasks:

- Unpacks the TAR file that contains the data collectors on the target platform.
- Schedules jobs to gather, format, and consolidate accounting data in the root account crontab file.
- On Red Hat Linux platforms, renames the file /etc/logrotate.d/psacct to /etc/logrotate.d/psacct.rpmsave.

- On SuSE Linux platforms, renames the file /etc/logrotate.d/acct to /etc/logrotate.d/acct.rpmsave.

Note: Installation of SmartCloud Cost Management Data Collectors for UNIX and Linux must be executed by the root account of the target platform to create directories and execute privileged commands.

The following sections describe how to remotely install SmartCloud Cost Management Data Collectors.

Note: The DB2 and Oracle database collectors require the operating system and file system data collectors. The operating system and file system data collectors must be installed on the target computer before the database collector. The Oracle and DB2 database collectors are provided in the IBM SmartCloud Cost Management Enterprise.

All remote installations require an installation file and a deployment manifest file. These files are located in <*SCCM_install_dir*>\collectors\unixlinux. The deployment manifest is an XML file that defines the installation file for the collectors. There are separate deployment manifest and installation files for the base collectors and the database collectors. The deployment manifest and installation files are further broken down by Linux or UNIX operating system. For example, for the AIX Power 64-bit platform, there are the following files: DeploymentManifest_aix_ppc64.xml and ituam_uc_aix_ppc64.tar (for the operating system and file system collectors) and DeploymentManifest_aix_ppc64_dblibs.xml and ituam_uc_dblibs_aix_ppc64.tar for the database collectors.

Prerequisites for the target computer when installing remotely

Make sure that the target computer meets the following prerequisites:

- The computer has .5 to 2 GB of available hard disk drive space depending on the process activity.
- SSH (Secure Shell) is running on the target computer and the computer is known to the SmartCloud Cost Management application server.

  Note: The Tectia SSH server is not supported as an SSH server on the target server.
- The UNIX/Linux process accounting subsystem is usually present on a platform. However, on some platforms, it must be installed separately. In this case, install the process accounting subsystem on your system. Examples of process accounting subsystems are:
  - For SUSE, rpm acct–6.3.5
  - For Red Hat, rpm psacct–6.3.2

Modify the sample deployment job file

SmartCloud Cost Management includes a sample job files that you can use to deploy UNIX and Linux data collectors from the SmartCloud Cost Management application server to multiple computers. After you have modified and optionally renamed these files, move the files to the <*SCCM_install_dir*>\jobfiles directory.

Table 145. Sample job files

Sample job file	Purpose
`SampleDeployLinuxCollector.xml`	Use this file to deploy the operating system and file system collectors.
`SampleDeployLinuxDatabaseCollector.xml`	Use this file to deploy the Oracle and DB2 database collectors. **Important:** The database collectors requires that the operating system and file system collectors are present on the target computer.

Deployment job file parameters

The following table describes the parameters that are specific to the sample deployment job files. Note that some optional parameters have default values. If you do not include these parameters or provide blank values, the default values are used. Sample deployment job files are provided in by Linux and UNIX platform in <*SCCM_install_dir*>\collectors\unixlinux.

Table 146. Deployment Job File Parameters

Parameter	Required or Optional	Description
`Action`	Required	Do not change this parameter.
`Host`	Required	The IP address or DNS name of the computer on which you want to install the collectors.
`UserId` `Password`	Required	The user ID must be the Super-User (root) account and the password must be the root password on the target UNIX or Linux server.
`KeyFilename`	Optional	You must use the SSH (Secure Shell) protocol to deploy the installation files. This parameter defines the SSH server's host key.
`Manifest`	Required	The deployment manifest file. Include the path only if the deployment manifest is in a location other than <*SCCM_install_dir*>\collectors\unixlinux on the SmartCloud Cost Management application server. The deployment manifest is an XML file that defines the installation file for the collectors. There are separate deployment manifest and installation files for the base collectors and the database collectors. The deployment manifest and installation files are further broken down by Linux or UNIX operating system. For example, for the AIX Power 64-bit platform, there are the following files: `DeploymentManifest_aix_ppc64.xml` and `DeploymentManifest_aix_ppc64_dblibs.xml`.

Table 146. Deployment Job File Parameters (continued)

Parameter	Required or Optional	Description
		The parameters in the deployment manifest also define the some of the default configuration options for the collectors. You can change the default parameter values in the deployment manifest or you can change the default values using the job file parameter `RPDParameters`. The parameters in the deployment manifest are the same as those defined in `RPDParameters`. If you do not want to use one or more of the default parameter values defined in the deployment manifest, it is recommended that you change the corresponding attribute value in `RPDParameters` rather than changing default values in the deployment manifest. For example, if you do not want to use the default value for the parameter `Parameter name="user"` that is defined in the deployment manifest, use the attribute `user=` in the job file parameter `RPDParameters` to define the user.
`Protocol`	Optional	The protocol used to deploy the installation files. To deploy to a Unix or Linux system, use `ssh` (Secure Shell). Make sure that SSH is enabled on the target computer.
`RPDParameters`	Optional	These are the same parameters that are in the deployment manifest. These parameters define some of the default configuration options for the collectors. The parameter values that you define here override the values in the deployment manifest. The parameters are: • `# path`-The full path in which you want to install the collectors. • `# user`-The account that owns the files on the target computer.

Table 146. Deployment Job File Parameters (continued)

Parameter	Required or Optional	Description
		• `cs_method`- This is the protocol that is used to transfer CSR files from the collector system to the SmartCloud Cost Management application server. Valid values are: `FTP` (file transfer protocol), `SCP` (secure copy, valid only if SmartCloud Cost Management application server server is on Windows), or `SFTP` (secure FTP, valid only if SmartCloud Cost Management application server server is on UNIX or Linux). The default value `HOLD` specifies that the files are held on the target computer. • `server`-This is the SmartCloud Cost Management application server name. • `cs_user`-This is the SmartCloud Cost Management application server account required to transfer CSR files from the collector system to the application server. • `cs_password`-This is the password for the account defined by the `cs_user` parameter. • `cs_proc_path`-This is the path to the `processes` directory on the SmartCloud Cost Management application server. If `processes` is on a UNIX or Linux, system, provide the full path to the directory. If `processes` is on a Windows system, provide the virtual directory that points the `processes` folder.
`Verbose`	Optional	The value `"true"` specifies that additional information is included in the job log file for debugging and troubleshooting purposes. If you set this parameter to `"false"`, do not include this parameter, or leave the parameter value blank, this additional information is not provided in the log file.
`SourcePath`	Required	The full path to the *<SCCM_install_dir>*`\collectors\unixlinux` folder on the SmartCloud Cost Management application server. This folder contains the installation files for the data collectors. The install file names are defined in the deployment manifest.

Table 146. Deployment Job File Parameters (continued)

Parameter	Required or Optional	Description
`UseSFTP`	Optional	If the `Protocol` parameter is set to `"ssh"`, use this parameter to specify whether the SFTP or SCP protocol will be used to transfer files. Valid values are: • `"true"` (the SFTP protocol is used) • `"false"` (the SCP protocol is used, this is the default) If you do not include this parameter or leave the parameter value blank, the SCP protocol is used. A value of `"true"` is used with certain SSH servers (such as Tectia SSH Server 5.x) to allow file transfers to complete successfully. This parameter is not shown in the sample deployment job files.

Running the deployment job file

To run the deployment job file from the command prompt, use the following command:

`startJobRunner.bat <deployment file name>.xml`

Or

`startJobRunner.sh <deployment file name>.xml`

Where *<deployment file name>* is the name that you give to the deployment job file.

To run the deployment job file from Administration Console:

1. In Administration Console, click **Task Management** > **Job Runner** > **Job Files**.
2. On the Job Runner Job Maintenance page, click the **View popup menu** icon for the job file that you want to run and then click **Run Job**.

Note: Make sure that the deployment file is in the *<SCCM_install_dir>*`\jobfiles` directory so that it will be displayed on the Job Runner Job Maintenance page.

Linux and UNIX data collection architecture

The following is an overview of the key components that comprise the architecture for the SmartCloud Cost Management data collectors that run on Linux and UNIX platforms.

Utilities

SmartCloud Cost Management include utilities that are used to set up, administer, and run the SmartCloud Cost Management data collectors that run on Linux and UNIX platforms. These utilities are in the `$ITUAM_UC_HOME/bin` directory. The files that the utilities access are in the `$ITUAM_UC_HOME/data` directory.

The following table lists each of the utilities used to set up the collectors and the files accessed by these utilities.

Table 147. Administration Utilities

Utility	Files Accessed
G_license	• A_setup.sys. Parameter file - contains license information.
A_setup	• A_holiday.sys. Holiday file. • A_imgmap.sys. Image/Package Mapping file. • A_dbinst.sys. Database Instance file. • A_queuemap.sys. Queue Mapping file. • A_setup.sys. Parameter file. • A_shift.sys. Shift file. • A_shift.tmp. Backup Shift file. • A_term_par.sys. Terminal Parameter file.
A_authorize	• A_uaf.sys. Authorization file.
A_login, A_login_xm, and A_switch	• A_activity.sys. Activity file. • A_uaf.sys. Authorization file. • A_validate.sys. Validation file.

Scripts

Data collectors use scripts to perform many operations including performing some setup steps and managing data collection and consolidation on a scheduled basis. Scripts are also available to assist in recovery if an error occurs. These scripts use the parameter files described in the next section to accomplish these tasks.

Scripts are in two locations: the $ITUAM_UC_HOME/etc directory and the $ITUAM_UC_HOME/scripts directory. The $ITUAM_UC_HOME/etc directory contains the scripts used for data collection.

The $ITUAM_UC_HOME/scripts directory contains subdirectories that group scripts by function. These scripts are described in detail in this guide as they apply to a particular feature or function.

Parameter Files

The following SmartCloud Cost Management parameter files are delivered so that a minimal Linux and UNIX data collection system operates without any tailoring for first time installations.

Parameter File	Description
Parameter File	The Parameter file ($ITUAM_UC_HOME/data/A_setup.sys) contains system-wide flags and parameters that define your configuration. SmartCloud Cost Management sets the initial values in the A_setup.sys file during installation. The file is created and maintained by the A_setup utility.

Parameter File	Description
Configuration Parameter File	The Configuration Parameter file (`$ITUAM_UC_HOME/data/A_config.par`) provides a common configuration file that is used by the data collection and consolidation scripts to define your SmartCloud Cost Management environment. You can modify the environment variables in this as needed for your organization using a text editor such as vi.
Node File	The Node Parameter file (`$ITUAM_UC_HOME/data/A_node.par`) contains one entry for the node on which the file is installed. For example, if the file is installed on the server zeus, then the node entry in the file is zeus.

Setting the sharable library path

Users who run one of the SmartCloud Cost Management Linux or UNIX data collector utilities directly must have their collector `bin` directory in their sharable library path.

If you run the Oracle or DB2 data collector on a Linux or UNIX, you are required to run the `A_setup` utility. This utility requires that your collector `bin` directory is in your sharable library path.

Note: The Oracle and DB2 database collectors are provided in IBM SmartCloud Cost Management Enterprise.

The name of the sharable library path variable is different on different UNIX/Linux machines:

- AIX LIBPATH
- HPUX SHLIB_PATH
- Linux ,Solaris LD_LIBRARY_PATH

To include the collector bin directory in the sharable library path (assume the collector is installed in `/opt/ibm/SCCM/collectors/unix`):

```
> LD_LIBRARY_PATH=/opt/ibm/SCCM/collectors/unix/bin  ;  export  LD_LIBRARY_PATH
```

Setting the environment variables for data collection and consolidation

The Configuration Parameter file (`$ITUAM_UC_HOME/data/A_config.par`) provides a common configuration file that is used by the scripts to define your SmartCloud Cost Management environment. The file is commented and organized by the stages in the data collection process. The initial variable values in the `A_config.par` file are set during installation. However, you can modify these values as needed for your organization using a text editor (for example, vi).

Note: When you upgrade to a new release of SmartCloud Cost Management, some environment variables in the `A_config.par` file are overwritten to the default values. A backup file configuration file named `A_config.bak` contains the variable settings prior to the upgrade. After you complete the upgrade, you should compare the values in the `A_config.par` file to the values in the `A_config.bak` file.

The following table describes the key variables in the A_config.par file. Descriptions of each variable are also provided in the file.

Table 148. Environment Variables in the A_config.par File

Variable	Description
SmartCloud Cost Management Directory Variables	
ITUAM_USER and ITUAM_UPATH	These variables define the directory paths used by the data collectors. These variables are defined during installation.
	• ITUAM_USER-This is the SmartCloud Cost Management user account on the collector platform. • ITUAM_UPATH-This is the root account's directory path. On most systems, this is / or /root.
Data Collection File Cleanup Variables	
CLEANUP_HISTORY and CLEANUP_AGE	The CLEANUP_HISTORY variable specifies if files in the $ITUAM_UC_HOME/history directory are purged. If set to Y, files older than the CLEANUP_AGE value are purged from the directory each night as part of the execution of the ituam_uc_nightly script. The default CLEANUP_HISTORY value is Y. The default CLEANUP_AGE value is +4. Files older than four days are purged.
CLEANUP_ACCT and CLEANUP_ACCT_AGE	The CLEANUP_ACCT variable specifies if in the $ITUAM_UC_HOME/accounting directory is purged. If set to Y, files older than the CLEANUP_ACCT_AGE value are purged from the directory each night as part of the execution of the ituam_uc_nightly script. The default CLEANUP_ACCT value is Y. The default CLEANUP_ACCT_AGE value is +45. Files older than 45 days are purged.
CLEANUP_CSIS and CLEANUP_CSIS_AGE	The CLEANUP_CSIS variable specifies if in the $ITUAM_UC_HOME/CS_input_source directory is purged. If set to Y, files older than the CLEANUP_ACCT_AGE value are purged from the directory each night as part of the execution of the ituam_uc_nightly script. The default CLEANUP_ACCT value is Y. The default CLEANUP_ACCT_AGE value is +45. Files older than 45 days are purged.
CLEANUP_CLIENT_ACC	This variable specifies whether nightly accounting and storage files are to be purged from the history directory after they have been transferred to the $ITUAM_UC_HOME/accounting/<nodename> directory. The default is Y.
CLEANUP_STATFILES	This variable specifies whether temporary accounting files created by the A_format utility are to be purged from the data directory. The default is Y.

Table 148. Environment Variables in the A_config.par File (continued)

Variable	Description
Data Collection File Transfer Variables	
TRANSFER_VIA, ITUAM_SERVER, and ITUAM_DEST	These variables are used to transfer the nightly accounting and storage files from the history directory to the $ITUAM_UC_HOME/accounting/<nodename> directory.
ITUAM_KEY	This variable is required only if TRANSFER_VIA is set to FTP. In this situation, set this variable to the password for the account designated by ITUAM_USER.
Data Collection for AIX Advanced Accounting Variables:	
AACCT_TRANS_IDS	This variable designates the AIX Advanced Accounting record types that are included in the usage reports created by the ituam_format_aacct script. Valid values are: • 1 Process record • 2 Aggregated process record • 4 System processor and memory interval record • 6 File system activity interval record • 7 Network interface I/O interval record • 8 Disk I/O Interval record • 10 Sever VIO interval record • 11 Client VIO interval record • 16 Aggregated ARM transaction record The default is "1,4,6,7,8".
AACCT_ONLY	Set this variable to Y if you want to collect AIX Advanced Accounting data and *do not* want to collect traditional UNIX/Linux process accounting data. The default is N.
Data Collection for Oracle Variables:	
A_ORACLE_ACCT	Set this variable to Y if you want to collect Oracle data. This variable instructs the ituam_uc_nightly script to include the SmartCloud Cost Management Oracle Accounting file ($ITUAM_UC_HOME/data/A_dbacct.sys) in the collection and formatting of the nightly accounting file. The default is N. Note that to include Oracle data in the CSR file, you must set the GEN_ORACLE variable to Y.
USE_SESSION_OSUSER	Set this variable to Y to instruct the SmartCloud Cost Management Oracle Accounting daemon to retrieve the OS user name from the V$SESSION table. By default, the daemon will retrieve the name from the V$PROCESS table. In some environments, a more unique value can be found the V$SESSION table. The default is Y.

Table 148. Environment Variables in the A_config.par File (continued)

Variable	Description
ORACLE_STR_SAMPLE	Set this variable to Y if you want to include Oracle tablespace and datafile storage data in the collection and formatting of the nightly accounting file. The default is N. Note that to include Oracle storage data in the CSR file, you must also set the GEN_ORACLE_STORAGE variable to Y.
ORA_SEND_STARTMSG	If you are using the ituam_check_odb script with the start argument, setting this variable to Y instructs the script to send notification via e-mail to the list of users specified by the ORA_STARTMSG_RCPT variable. The default is N.
ORA_STARTMSG_RCPT	Set this variable to the list of e-mail users to be notified if the SmartCloud Cost Management Oracle Accounting daemon is re-started by the ituam_check_odb script. Use a comma to separate multiple e-mail addresses. This variable is valid only if the ORA_SEND_STARTMSG variable is set to Y. The default is N.
TNS_ADMIN	Set this variable to the location of the tnsnames.ora file if other than the $ORACLE_HOME/network/admin directory.
Data Collection for DB2 Variables	
A_DB2_ACCT	Set this variable to Y if you want to collect DB2 data. This variable instructs the ituam_uc_nightly script to include the SmartCloud Cost Management DB2 Accounting file ($ITUAM_UC_HOME/data/A_db2acct.sys) in the collection and formatting of the nightly accounting file. The default is N. Note that to include DB2 data in the CSR file, you must also set the GEN_DB2 variable to Y.
DB2_STR_SAMPLE	Set this variable to Y if you want to include DB2 database partition storage information in the collection and formatting of the nightly accounting file. The default is N. Note that to include DB2 storage data in the CSR file, you must also set the GEN_DB2_STORAGE variable to Y.
DB2_SEND_STARTMSG	If you are using the ituaml_check_db2 script with the start argument, setting this variable to Y instructs the script to send notification via e-mail to the list of users specified by the DB2_STARTMSG_RCPT variable. The default is N.
DB2_STARTMSG_RCPT	Set this variable to the list of e-mail users to be notified if the SmartCloud Cost Management DB2 Accounting daemon is re-started by the ituam_check_db2 script. Use a comma to separate multiple e-mail addresses. This variable is valid only if the DB2_SEND_ STARTMSG variable is set to Y.
Data Collection for Storage Variables	

Table 148. Environment Variables in the A_config.par File (continued)

Variable	Description
ITUAM_SAMPLE	Set this variable to Y if you want to collect file system storage data. This data is provided in the Storage file ($ITUAM_UC_HOME/data/A_storage.sys). The default is Y.
ITUAM_DYNAMIC_STORAGE_PAR	If this variable and ITUAM_SAMPLE are both set to Y, the Storage Parameter file ($ITUAM_UC_HOME/data/A_storage.par) is rebuilt each night before the sampler script is run. The default is Y.
Data Consolidation Variables for proc_multi Script	
ITUAM_NODE_FILE	This variable defines the Node Parameter file. The default file is $ITUAM_UC_HOME/data/A_node.par.
SELECT_QUALS	This variable is used to pass command line qualifiers to the Select utility ($ITUAM_UC_HOME/bin/A_select) when the utility is called from the proc_multi script. For example, if PROPRIETARY_SOFTWARE is enabled in the $ITUAM_UC_HOME/data/A_setup.sys file and you want to automatically add process names to the Image Mapping file ($ITUAM_UC_HOME/data/A_image.sys), define this variable as follows: SELECT_QUALS=/ADD_IMAGE (this is the default)
RANGE_BACK and RANGE_AHEAD	These variables define a window for consolidating data. The default for both variables is 3.
Data Consolidation Variables for Generating CSR Files	
GEN_UNIXPROC	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX process usage data. The default is Y.
GEN_PROCONLY	If set to Y, interactive, background, and storage type records are not included in the CSR file. The usage in these types of records is already included in the process and file system type records. The default is Y.
GEN_UNIXFS	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX file system usage data. The default is Y.
GEN_ORACLE	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX Oracle usage data. The default is N.

Table 148. Environment Variables in the A_config.par File (continued)

Variable	Description
GEN_ORACLE_STORAGE	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX Oracle tablespace and data file utilization data. The default is N.
GEN_DB2	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX DB2 usage data. The default is N.
GEN_DB2_STORAGE	If set to Y, this variable instructs the CS_nightly_consolidation script to create a CSR file containing UNIX DB2 partition storage utilization data. The default is N.
Variables for Sending CSR Files to the SmartCloud Cost Management Application Server	
CS_PLATFORM through CS_PROC_PATH	These variables are used to transfer the CSR files to the SmartCloud Cost Management application server.
Other variables	
TURN_WTMP	If this variable is set to Y, a new UNIX or Linux wtmp (or wtmpx) file is created each night when the ituam_uc_nightly script executes. The default is N.

Setting up operating and file system data collection: starting Linux or UNIX process accounting

If you installed the operating system and file system data collectors remotely, process accounting is started automatically. The following steps are required only if you want to run the operating system and file system data collectors on the SmartCloud Cost Management application server, where they were installed with the SmartCloud Cost Management product.

To start process accounting:

1. SmartCloud Cost Management manages the Linux or UNIX process accounting file, var/account/pacct. Therefore, you should verify that no other processes are currently manipulating this file.
2. On most UNIX platforms, check in the adm account crontab file to verify that the accounting scripts runacct, monacct, and ckpacct are not currently scheduled. If these scripts are scheduled, comment them out of the adm account crontab file.
3. On Linux platforms, the cron.daily script calls the logrotate script, which compresses the current process accounting file, pacct, and starts a new file. You must disable this function as follows:
   - On Red Hat Linux platforms, copy the script /etc/logrotate.d/psacct to /etc/logrotate.d/psacct.rpmsave.

- On SuSE Linux platforms, copy the script `/etc/logrotate.d/acct` to `/etc/logrotate.d/acct.rpmsave`.

4. Start UNIX/Linux process accounting using the `$ITUAM_UC_HOME/etc/turnacct` script. As the root user, execute the following command:

   ```
   > $ITUAM_UC_HOME/etc/turnacct on
   ```

   If you need to suspend UNIX/Linux process accounting, call this script with the argument `off`.

Metrics Collected

The following metrics are collected from the `wtmp` file:

- User name of the user that logged in.
- Controlling terminal (when there is one).
- Name of the remote system (when there is one).
- Time logged in or logged out.

Note: Most UNIX systems do not store the user name in the logout record, only in the login record.

The following metrics collected from the `pacct` file:

- Command name.
- UID and GID of the user that executed the command.
- Controlling terminal (when there is one).
- Amount of user and system CPU time that it took to execute that command.
- Time the command started executing.
- Time that it took to execute the command (the *elapsed* time).
- Average memory usage.
- Number of blocks read or written.
- Where provided by the system, the number of characters transferred.

Identifiers and resources defined from process accounting data (operating system and file system)

This topic describes the identifiers and resources that are defined by the Operating System and File System collectors.

By default, the following data collected by the Operating System and File System collectors are defined as chargeback identifiers and resource rate codes. The rate codes assigned to the resources are pre-loaded in the Rate table.

Operating system identifiers and resource rate codes

Table 149. UNIX Operating System Identifiers

Identifier	Description
Software Package Identifiers	
SYSTEM_ID	Server node name.
USERNAME	OS user name.
PROCESSNAME	Process name.

Table 149. UNIX Operating System Identifiers (continued)

Identifier	Description
Interactive, Background, and Storage Identifiers	
SYSTEM_ID	Server node name.
USERNAME	OS user name.

Table 150. UNIX Operating System Rate Codes

Rate Code	Description
Software Package Rate Codes	
LLG101	UNIX Process Block I/O (1,000s)
LLG102	UNIX Process Character I/O (100,000s)
LLG103	UNIX Process Image Time (Hours)
LLG104	UNIX Process User CPU (Minutes)
LLG105	UNIX Process System CPU (Minutes)
LLG106	UNIX Process Total CPU (Minutes)
LLG107	UNIX Process Memory (MB Days)
LLG108	UNIX Process Image Count
LLG109	UNIX Process SU Image Count
LLG110	UNIX Process Chg Image Time (Hours)
Interactive Rate Codes	
LLA101	UNIX Interactive Block I/O (1,000s)
LLA102	UNIX Interactive Character I/O (100,000s)
LLA103	UNIX Interactive Image Time (Hours)
LLA104	UNIX Interactive Connect Time (Hours)
LLA105	UNIX Interactive User CPU (Minutes)
LLA106	UNIX Interactive System CPU (Minutes)
LLA107	UNIX Interactive Total CPU (Minutes)
LLA108	UNIX Interactive Memory (MB Days)
LLA109	UNIX Interactive Image Count
LLA110	UNIX Interactive Logins
LLA111	UNIX Interactive SU Image Count
LLA112	UNIX Interactive SU Count
LLA113	UNIX Interactive SU Time (Hours)
LLA114	UNIX Interactive Window Time (Hours)
LLA115	UNIX Interactive Chg Image Time (Hours)
LLA116	UNIX Interactive Chg Connect Time (Hours)
LLA117	UNIX Interactive Chg SU Time (Hours)
LLA118	UNIX Interactive Chg Win Time (Hours)

Table 150. UNIX Operating System Rate Codes (continued)

Rate Code	Description
Background Rate Codes	
LLB101	UNIX Background Block I/O (1,000s)
LLB102	UNIX Background Character I/O (100,000s)
LLB103	UNIX Background Image Time (Hours)
LLB104	UNIX Background User CPU (Minutes)
LLB105	UNIX Background System CPU (Minutes)
LLB106	UNIX Background Total CPU (Minutes)
LLB107	UNIX Background Memory (MB Days)
LLB108	UNIX Background Image Count
LLB109	UNIX Background Logins
LLB110	UNIX Background Chg Image Time (Hours)
Storage Rate Codes	
LLD101	UNIX Block Weeks (not in CIMSRate table by default)

File system identifiers and resource rate codes

Table 151. File System Identifiers

Identifier	Description
SYSTEM_ID	Server node name.
FS_MOUNT_PT	File system mount point.
FS_DEVICENAME	File system device name.

Table 152. File System Rate Codes

Rate Code	Description
LLR101	UNIX Filesystem Size (512-byte Blocks)
LLR102	UNIX Filesystem Blocks Used (512-byte Blocks)
LLR103	UNIX Filesystem Number of Files
LLR104	UNIX Filesystem Size (GB Days)
LLR105	UNIX Filesystem Used (GB Days)

Setting up AIX Advanced Accounting data collection

This section describe how to set up Advanced Accounting data collection on a AIX system. To collect Advanced Accounting data, the system must be at AIX 5.3, Maintenance Level 3 or later.

Installing the UNIX/Linux AIX Advanced Accounting Collector

This topic describes how to install the UNIX/Linux AIX Advanced Accounting Collector.

About this task

The SmartCloud Cost Management UNIX/Linux Collector can be installed in one of two ways, using RXA Deployment from SmartCloud Cost Management Server or by installing manually on the AIX platform. When using either method, be sure to include the **aacct_config=TRUE** parameter.

Installing the UNIX/Linux AIX Advanced Accounting Collector Manually: Procedure

1. Copy the following files from the SmartCloud Cost Management Server to the directory on the target AIX platform where you want the collector to be installed:
   a. `..\collectors\unixlinux\tuam_unpack_uc_collector --> /opt/tuam/collectors/unix/tuam_unpack_uc_collector`
   b. `..\collectors\unixlinux\ituam_uc_aix5.tar --> /opt/tuam/collectors/unix/ituam_uc_aix5.tar`
2. As root user change to the UNIX directory using the command:**cd /opt/tuam/collectors/unix/**
3. Use the following command to install the collector:
   ./tuam_unpack_uc_collector path=/opt/tuam/collectors/unix user=root cs_method=FTP server=TUAM_SERVER cs_user=cadmin cs_pwd=cadminpasswd aacct_config=TRUE

Installing the UNIX/Linux AIX Advanced Accounting Collector using RXA deployment:
Procedure

1. Install the SmartCloud Cost Management UNIX/Linux Collector using the `SampleDeployUnixLinuxCollector.xml` jobfile.
2. See the SmartCloud Cost Management Documentation on installing the UNIX/Linux Core Collector. You will probably want to set the RPDParameters to something like the following: `<Parameter RPDParameters = "path=/opt/tuam/collectors/unix ;user=root ;cs_method=FTP ;server=TUAM_SERVER ;cs_user=cadmin ;cs_pwd=cadminpasswd ;aacct_config=TRUE ;"/>`

Post Install Configuration:
About this task

If Advanced Accounting has not been configured previously, follow these steps:

Procedure

1. As root user, enter the following commands:
   a. **acctctl fadd /var/aacct/aacct1.dat 10**
   b. **acctctl fadd /var/aacct/aacct2.dat 10**
   c. **acctctl isystem 5**
   d. **acctctl iprocess 5**
   e. **acctctl agproc on**
   f. **acctctl agarm on**

g. **acctctl agke on**

h. **acctctl on**

2. The following settings need to be made in UNIX/Linux Collector configuration:
   a. In /opt/tuam/collectors/unix/data/A_config.par, CS_COLL_PATH is used when sending the nightly AACCT data files to the SmartCloud Cost Management Server. Set CS_COLL_PATH to the path to the ...\logs\collectors folder on the SmartCloud Cost Management Server.
   b. In /opt/tuam/collectors/unix/scripts/enterprise/CS_log_send, set ADD_IP=Y to include the ip address in the folder name on the SmartCloud Cost Management Server where the nightly AACCT files are delivered.
3. On the SmartCloud Cost Management Server, be sure the AACCT_4 folder has been created in <SCCM_install_dir>\logs\collectors. The AACCT nightly files are sent using the CS_log_send script called from ituam_send_aacct. This script does not create the .../logs/collectors/AACCT_4 folder. The script will create subfolders for the clients.
4. If there are machines on which the SmartCloud Cost Management UNIX/Linux collector had been installed before and configured for collecting traditional UNIX Process accounting, (pacct). You will want to be sure this is turned off. You can do this by running the following command. If you are unsure, its ok to run this command anyway.**/opt/tuam/collectors/unix/etc/turnacct off**

Creating Advanced Accounting data files

Using the Advance Accounting utility acctctl, create two data files with enough space to hold a day of Advanced Accounting records. The space required depends on the amount of process activity and the Advanced Accounting configuration.

Advanced Accounting collects and can aggregate accounting information as determined by its configuration. If Advanced Accounting is configured to aggregate processing records, a smaller file size is required. If Advanced Accounting is not configured to aggregate records, a larger file size is required. Aggregated records do not include the process name in the record.

To create the Advanced Accounting Data files, use these commands:

```
> acctctl fadd /var/aacct/aacct1.dat 4

> acctctl fadd /var/aacct/aacct2.dat 4
```

These commands create two data files allocated for 4 MB each. This is enough space for a day of hourly system interval records as well as aggregated process records.

Configuring AIX Advanced Accounting

Configure Advance Accounting to generate system interval and aggregated process records as shown in these example commands:

```
> acctctl isystem 60

> acctctl iprocess 60
```

In this example, the system interval records are created every 60 minutes. The process interval is also set to 60 minutes. The process interval record time is required because process aggregation is set to on.

The command `agproc on` enables record aggregation. If you do not want to aggregate records (for example, you want accounting information at the process name level), use the command `agproc off`. If you do not use aggregation, you must create larger data files. The following are the command for agproc on and off:

```
> acctctl agproc on

> acctctl agproc off
```

If you are gathering ARM application transactions or third-party kernel extension records, the data files might require a larger size and you must enable aggregation of these records as follows:

```
> acctctl agarm on

> acctctl agke on
```

Setting up Advanced Accounting data collection

Advanced Accounting data can be collected manually or automatically.

Advanced Accounting data can be collected manually or automatically using the following scripts in the `$ITUAM_UC_HOME/scripts/aacct` directory. These scripts use values in the `$ITUAM_UC_HOME/data/A_config.par` file. If you want to collect Advanced Accounting data and do not want to collect traditional UNIX/Linux process accounting, set the variable `AACCT_ONLY=Y` in the `A_config.par` file.

- **ituam_get_aacct.** This script retrieves the "Active" Advanced Accounting data file and copies it to a `ITUAM_UC_HOME/history/aacct<date>.dat` file. The script can be called multiple times a day if needed. If this script is called more than once a day, subsequent Advanced Accounting data files are copied to `ITUAM_UC_HOME/history/aacct_n_<date>.dat`, where `n` specifies the sequential numbering of the data files.

  After copying the file the script restarts accounting and resets the AACCT data file. The ability to execute the script multiple times a day is provided as a means to respond to alerts from Advanced Accounting when data files are nearing capacity. This should not occur if data files have been created with enough space to hold data for an entire day.
- **ituam_format_aacct.** This script generates usage logs from the Advanced Accounting data files in the `history` directory (`ITUAM_UC_HOME/history/aacct_n_<date>.dat`).

  The Advanced Accounting record types that are included in the logs are defined by the `AACCT_TRANS_IDS` variable in the `A_config.par` file. By default, `AACCT_TRANS_IDS` is set to `"1,4,6,7,8"`. Each ID number indicates one of the following Advanced Accounting Record types:
  - 1 or 2 Process records
  - 4 System processor and memory interval record
  - 6 File system activity interval record
  - 7 Network interface I/O interval record
  - 8 Disk I/O Interval record
  - 10 Virtual I/O Server interval record
  - 36 WPAR System interval record
  - 38 WPAR File System activity interval record
  - 39 WPAR Disk I/O interval record
  -

Note: If you have the ITUAM agent installed and running on the Virtual I/O Server, you can generate data files that contain record type 10 directly on the Virtual I/O Server.

- 11 Virtual I/O Client interval record
- 16 Aggregated ARM transaction record

The `ituam_format_aacct` script creates the usage logs in the `$ITUAM_HOME/CS_input_source/aacctn_<date>.txt`, where `n` is and Advanced Accounting record type.

You can call the `ituam_format_aacct` script with an optional date argument where date is in the format YYYYMMDD. The date argument specifies that only those data files that contain the specified date in the file name are included in the usage report/log. If no argument is present, data files with the current date in the file name are included. If multiple data files exist in the `history` directory for the specified day, all files for that day are included in the generated logs.

Examples

`ituam_format_aacct` (Advanced Accounting records with the current date are included in the usage logs)

`ituam_format_aacct 20070916` (Advanced Accounting records with the date 20070916 are included in the usage logs)

- **ituam_send_aacct.** This script is used to transport the usage logs to the SmartCloud Cost Management application server where they are processed and loaded into the SmartCloud Cost Management database.

  You can call the script with an optional date argument where date is in the format YYYYMMDD. The date argument specifies that only those usage logs that contain the specified date in the file name are transferred. If no argument is present, logs with the current date in the file name are transferred.

  The `ituam_send_aacct` script uses the`$ITUAM_HOME/scripts/enterprise/CS_log_send` to connect and transfer the logs to the SmartCloud Cost Management server. Refer to the notes in the `A_config.par` file to determine the settings that work best for your environment.

  The `CS_log_send` script uses the following variables in the `A_config.par` file.

  - `CS_METHOD` .This is the protocol that is used to transfer CSR files from the collector system to the SmartCloud Cost Management application server. Valid values are: `RCP` (remote copy protocol), `FTP` (file transfer protocol), `SCP` (secure copy), or `SFTP` (secure FTP).
  - `CS_USER`. This is the account required to log on to the SmartCloud Cost Management server.
  - `CS_KEY`. This variable is required only if the `TRANSFER_VIA` is set to `FTP`. In this situation, set this variable to the password for the account designated by `CS_USER`.
  - `CS_COLL_PATH`. This is the path for the `logs/collectors` directory.

    If `logs/collectors` is on a UNIX or Linux, system, provide the full path to the directory.

    If `logs/collectors` is on a Windows system, provide the virtual directory that points the `logs/collectors` directory.
  - `CS_PLATFORM`. The SmartCloud Cost Management application server name.
  - `CS_UPATH`. If the `CS_METHOD` is `FTP`, the home directory for the account running the `CS_log_send` script.

Scheduling Advanced Accounting data collection

Jobs to collect and format Advanced Accounting data should be scheduled in root account crontab.

The following example crontab entries collect the current AACCT data files, extract usage logs, and send the logs to the SmartCloud Cost Management application server where they are processed and loaded into the SmartCloud Cost Management database.

```
5 1 * * * ( /opt/IBM/tivoli/ituam/collectors/unix/scripts/aacct/ituam_get_aacct 1>
 /opt/IBM/tivoli/ituam/collectors/unix/log/ituam_get_aacct.log 2>&1 )
10 1 * * * ( /opt/IBM/tivoli/ituam/collectors/unix/scripts/aacct/ituam_format_aacct 1>
 /opt/IBM/tivoli/ituam/collectors/unix/log/ituam_format_aacct.log 2>&1 )
15 1 * * * ( /opt/IBM/tivoli/ituam/collectors/unix/scripts/aacct/ituam_send_aacct 1>
 /opt/IBM/tivoli/ituam/collectors/unix/log/ituam_send_aacct.log 2>&1 )
```

Advanced Accounting metrics collected and SmartCloud Cost Management rate codes

The tables in the following sections show the metrics that are collected from the Advanced Accounting data files by record type. The table also shows the SmartCloud Cost Management rate code that is assigned to each metric.

Process Metrics Collected (Record Type 1)

Table 153. Advanced Accounting Process Metrics Collected

Metric	Rate Code
Interval count	**AAID0101**
Elapsed time in seconds	**AAID0102**
Elapsed thread time in seconds	**AAID0103**
CPU time in seconds	**AAID0104**
Elapsed disk pages in seconds	**AAID0105**
Elapsed real pages in seconds	**AAID0106**
Elapsed virtual memory pages in seconds	**AAID0107**
Local file I/O in MB	**AAID0108**
Other file I/O in MB	**AAID0109**
Local sockets I/O in MB	**AAID0110**
Other sockets I/O in MB	**AAID0111**

System Metrics Collected (Record Type 4)

Table 154. Advanced Accounting System Metrics Collected

Metric	Rate Code
Number of CPUs (interval aggregate)	**AAID0401**
Entitled capacity (interval aggregate)	**AAID0402**
System pad length (interval aggregate)	**AAID0403**
System idle time in seconds	**AAID0404**
User process time in seconds	**AAID0405**
Interrupt time in seconds	**AAID0406**
Memory size in MB (interval aggregate)	**AAID0407**
Large page pool in MB	**AAID0408**
Large page pool in-use in MB	**AAID0409**

Table 154. Advanced Accounting System Metrics Collected (continued)

Metric	Rate Code
Pages in	**AAID0410**
Pages out	**AAID0411**
Number start I/O	**AAID0412**
Number page steals	**AAID0413**
I/O wait time in 1/1000 seconds	**AAID0414**
Kernel process time in 1/1000 seconds	**AAID0415**
Interval elapsed time in 1/1000 seconds	**AAID0416**
Percent User CPU	**AAID0417**[1]
Percent System CPU	**AAID0418**[1]
Percent Idle CPU	**AAID0419**[1]
Percent I/O Wait	**AAID0420**[1]
Percent Interupt	**AAID0421**[1]
Number/Amount Physical CPU	**AAID0422**[1]
AIX System Amount of physical memory allocated to a partition in page-seconds (in thousands)	**AAID0423**[1]
AIX System Amount of entitled memory allocated to a partition reported in 4K pages	**AAID0424**[1]
AIX System Amount of I/O memory entitlement used by a partition in page-seconds (in thousands)	**AAID0425**[1]

[1]These rate codes are not pre-loaded in the Rate table. To use these rate codes, you can load the codes into the Rate table using Administration Console. To convert these metrics from 1/1000 seconds to seconds, an Integrator stage can be added to the job file used to process type 4 records as shown in the following example:

```
<Stage name="ResourceConversion" active="true">

<Resource name="AAID0415">

<FromResources>

<FromResource name="AAID0415" symbol="a"/>

</FromResources>

</Resource>

</Resources>

<Parameters>

<Parameter formula="a/1000"/>

</Parameters>

</Stage>
```

File System Metrics Collected (Record Type 6)

Table 155. Advanced Accounting File System Metrics Collected

Metric	Rate Code
Bytes transferred in MB	**AAID0601**
Read/write requests	**AAID0602**
Number opens	**AAID0603**
Number creates	**AAID0604**
Number locks	**AAID0605**

Network Metrics Collected (Record Type 7)

Table 156. Advanced Accounting Network Metrics Collected

Metric	Rate Code
Network number I/O	**AAID0701**
Network bytes transferred in MB	**AAID0702**

Disk Metrics Collected (Record Type 8)

Table 157. Advanced Accounting Disk Metrics Collected

Metric	Rate Code
Transfers	**AAID0801**
Block reads	**AAID0802**
Block writes	**AAID0803**
Transfer block size (interval aggregate)	**AAID0804**

Virtual I/O Server Metrics Collected (Record Type 10)

Table 158. Advanced Accounting Virtual I/O Server Metrics Collected

Metric	Rate Code
Server bytes in	**AAID1001**
Server bytes out	**AAID1002**

Virtual I/O Client Metrics Collected (Record Type 11)

Table 159. Advanced Accounting Virtual I/O Client Metrics Collected

Metric	Rate Code
Client bytes in	**AAID1101**
Client bytes out	**AAID1102**

ARM Transaction Metrics Collected (Record Type 16)

Table 160. Advanced Accounting ARM Transaction Metrics Collected

Metric	Rate Code
Application time	**AAID1601**
Response time in seconds	**AAID1602**

Table 160. Advanced Accounting ARM Transaction Metrics Collected (continued)

Metric	Rate Code
Queued time in seconds	**AAID1603**
Application CPU time in seconds	**AAID1604**

WPAR System Metrics Collected (Record Type 36)

Table 161. Advanced Accounting WPAR System Metrics Collected

Metric	Rate Code
Number of CPUs (Global interval value)	**AAID3601**
Entitled capacity (Global interval value)	**AAID3602**
System pad length (Global interval value)	**AAID3603**
System idle time (Unused)	**AAID3604**
User process time in seconds	**AAID3605**
Interrupt time (Unused)	**AAID3606**
Memory size in MB (Global interval value)	**AAID3607**
Large page pool in MB (Unused)	**AAID3608**
Large page pool in-use in MB (Unused)	**AAID3609**
Pages in	**AAID3610**
Pages out	**AAID3611**
Number start I/O	**AAID3612**
Number page steals	**AAID3613**
I/O wait time (Unused)	**AAID3614**
Kernel process time in 1/1000 seconds	**AAID3615**
Interval elapsed time in 1/1000 seconds	**AAID3616**

WPAR File System Metrics Collected (Record Type 38)

Table 162. Advanced Accounting WPAR File System Metrics Collected

Metric	Rate Code
Bytes transferred in MB	**AAID3801**
Read/Write requests	**AAID3802**
Number opens	**AAID3803**
Number creates	**AAID3804**
Number locks	**AAID3805**

WPAR Disk I/O Metrics Collected (Record Type 39)

Table 163. Advanced Accounting WPAR Disk I/O Metrics Collected

Metric	Rate Code
Network number I/O	**AAID3901**
Network bytes transferred in MB	**AAID3902**

Setting up Virtual I/O Server data collection

The Virtual I/O Server includes several IBM Tivoli agents and clients, including the ITUAM agent. The ITUAM agent is packaged with the Virtual I/O Server and is installed when the Virtual I/O Server is installed. If the ITUAM agent is run on the Virtual I/O Server, the agent produces a data file from which usage data is collected.

Note: SmartCloud Cost Management supports the ITUAM agent on the Virtual I/O Server. However, SmartCloud Cost Management and the ITUAM agent are separate products. For more information about the Virtual I/O Server and the ITUAM agent, refer to the *System p Advanced Power Virtualization Operations Guide*.

About the Configuration Parameter file on the Virtual I/O Server

The Configuration Parameter file (`A_config.par`) provides a common configuration file that is used by scripts to define your SmartCloud Cost Management environment.

The `A_config.par` file on the Virtual I/O Server provides the environment variables required to run the ITUAM agent. The following table shows variables in the `A_config.par` file on the Virtual I/O Server that have *different* default values than the same variables in the `A_config.par` on the SmartCloud Cost Management application server.

The padmin user can edit the `A_config.par` file on the Virtual I/O Server (`$ITUAM_UC_HOME/data/A_config.par`). However, modifying the `A_config.par` file is usually not required unless you want to push files to the SmartCloud Cost Management application server or you want to change the value of the `AACCT_TRANS_IDS` variable.

Table 164. Environment Variables in the A_config.par File

Variable	Description
ITUAM_SAMPLE	This variable is set to `N` by default. Therefore, file system storage data is collected.
AACCT_TRANS_IDS	This variable designates the record types that is included in the log generated by the SmartCloud Cost Management agent. Valid values are any of the following: 1, 4, 6, 7, 8, 10, 11, or 16. The default is 10, Virtual I/O Server interval record.
AACCT_ONLY	This variable is set to `Y` by default. Therefore, traditional UNIX/ Linux process accounting data *is not* collected.

Configuring and starting the SmartCloud Cost Management Agent:

This topic describes how to configure the ITUAM agent.

About this task

To configure and start the ITUAM agent, complete the following steps:

Procedure

1. List all of the available ITUAM agents using the `lssvc` command. For example:

   ```
   $lssvc
   ITUAM_base
   ```

   In this example, `ITUAM_base` is the only ITUAM agent.
2. Using the `cfgsvc` command, list the attributes that are associated with the ITUAM agent that you want to configure as shown in the following example:

   ```
   $cfgsvc –ls ITUAM_base
   ACCT_DATA0
   ACCT_DATA1
   ISYSTEM
   IPROCESS
   ```
3. Configure the ITUAM agent with its associated attributes using the `cfgsvc` command:

   ```
   cfgsvc ITUAM_agent_name -attr ACCT_DATA0=value1 ACCT_DATA1=value2 ISYSTEM=value3 IPROCESS=value4
   ```

   Where:

   - *ITUAM_agent_name* is the name of the ITUAM agent. For example, `ITUAM_base`.
   - *value1* and *value2* are the size (in MB) of the data files that hold the daily accounting information. The space required depends on the amount of process activity and the ITUAM agent configuration.
   - *value3* is the time (in minutes) when the agent generates system interval records.
   - *value4* is the time (in minutes) when the system generates aggregate process records.

   Example:

   ```
   cfgsvc ITUAM_base -attr ACCT_DATA0=4 ACCT_DATA1=4 ISYSTEM=60 IPROCESS=60
   ```

   This command creates two data files allocated for 4 MB each. System interval records are created every 60 minutes and aggregate process records are also created every 60 minutes.
4. Start the ITUAM agent using the `startsvc` command. For example:

   ```
   startsvc ITUAM_base
   ```

   Note: If you want to stop the ITUAM agent, for example, to change the ITUAM agent configuration, the command is `stopsvc <agent name>`. Using the preceding example, the command would be `stopsvc ITUAM_base`.

Collecting and Converting Virtual I/O Server data files:

The data files that are created by the ITUAM agent are collected and formatted into a data file. The collection and formatting process is performed automatically using the scripts described in this section. These scripts run automatically from root's crontab using the values in the `A_config.par` file on the Virtual I/O Server.

The collection and formatting scripts are in the `$ITUAM_UC_HOME/scripts/aacct` directory on the Virtual I/O Server.

Using the ITUAM Agent collection and formatting scripts

The following scripts collect the Virtual I/O Server data file and generate usage logs. These logs are then transferred to the SmartCloud Cost Management application server where they are processed and loaded into the SmartCloud Cost Management database.

- **ituam_get_aacct**. This script retrieves the "Active" Virtual I/O Server data file and copies it to a `ITUAM_UC_HOME/history/aacct<date>.dat` file. The script can be called multiple times a day if needed. If this script is called more than once a day, subsequent Virtual I/O Server data files are copied to `ITUAM_UC_HOME/history/ aacct_n_<date>.dat`, where *n* specifies the sequential numbering of the data files.

  After copying the file, the script restarts accounting and resets the AACCT data file. The ability to execute the script multiple times a day is provided as a means to respond to alerts from the Virtual I/O Server when data files are nearing capacity. This should not occur if data files have been created with enough space to hold data for an entire day.
- **ituam_format_aacct**. This script generates usage logs from the Virtual I/O Server data files in the `history` directory (`ITUAM_UC_HOME/history/aacct_n_<date>.dat`).

  The Advanced Accounting record types that are included in the logs are defined by the `AACCT_TRANS_IDS` variable in the `A_config.par` file. By default, `AACCT_TRANS_IDS` is set to "10"

  The `ituam_format_aacct` script creates the usage logs in the `$ITUAM_HOME/CS_input_source/aacct10_<date>.txt`.

  You can call the `ituam_format_aacct` script with an optional date argument where date is in the format YYYYMMDD. The date argument specifies that only those data files that contain the specified date in the file name are included in the usage log. If no argument is present, data files with the current date in the file name are included. If multiple data files exist in the `history` directory for the specified day, all files for that day are included in the generated logs.

  Examples

  `ituam_format_aacct` (Virtual I/O Server records with the current date are included in the usage logs)

  `ituam_format_aacct 20070916` (Virtual I/O Server records with the date 20070916 are included in the usage logs)

Transferring Virtual I/O Server usage logs to the SmartCloud Cost Management application server:

To enable SmartCloud Cost Management to process the data that is in the Virtual I/O Server usage logs, the logs must be transferred to the SmartCloud Cost Management application server.

To transfer the logs, you can do either of the following:

- "Pull" the logs from Virtual I/O Server to the SmartCloud Cost Management application server. This method requires that you set up a job file on your Windows or UNIX platform to pull the file to the SmartCloud Cost Management application server for processing. This is the recommended method because the transfer process, including any failed transfers, is included in the nightly job log.
- "Push" the logs from the Virtual I/O Server to the SmartCloud Cost Management application server. This method requires that you set the file transfer variables in the `A_config.par` on the Virtual I/O Server

Pulling the Usage Logs To the SmartCloud Cost Management Server

To pull the usage logs to the SmartCloud Cost Management application server, create an XML job file on the application server to transfer the logs from the Virtual I/O Server.

Pushing the Usage Logs To the SmartCloud Cost Management Server

Note: Pushing the usage logs to the SmartCloud Cost Management application server is not recommended because you must configure the `A_config.par` file on the Virtual I/O Server. It is recommended that you pull the log files from the SmartCloud Cost Management application server.

You can use the `ituam_send_aacct` script to transfer the usage logs to the application server. To use this script, you must uncomment the contab entry that calls the script in the root account crontab file.

The logs are placed in the `logs/collector` directory on the application server in the following subdirectory architecture: `$ITUAM_HOME\collectors\AACCT_n\<feed>`, where *n* specifies the record type and feed specifies the log file source. For example, `$ITUAM_HOME\collectors\AACCT_10\vioray` specifies that usage logs containing record type 10 from the Virtual I/O Server named `vioray` are stored in this path.

To push the usage logs to the SmartCloud Cost Management application server:

- Set the file transfer parameters in the `A_config.par` file on the Virtual I/O Server.
- Run the `ituam_send_aacct` script from root's crontab.

Configuring the File Transfer Variables in the A_config.par File on the Virtual I/O Server

To transfer the usage logs to the SmartCloud Cost Management application server using the `ituam_send_aacct` script, the padmin user must set the following variables in the `A_config.par` file on the Virtual I/O Server.

The variable descriptions are grouped by `CS_METHOD` variable value because this value determines the values that you can set for other variables. The `CS_METHOD`

variable defines the protocol that is used to transfer the usage logs to the SmartCloud Cost Management application server.

Table 165. Variables for File Transfer to SmartCloud Cost Management application server

Variable	Description
CS_METHOD=FTP	
CS_USER	The account required to log on to the Accounting Manager application server. If the application server is on a Windows system and you are using a domain account, use the following format for the user name: CS_USER="<domain name>\\\\<account name>" Make sure that you use the four backslashes between the account name and domain name as shown.
CS_KEY	The password for the account defined by the CS_USER variable.
CS_COLL_PATH	The path to the logs/collectors directory on the SmartCloud Cost Management application server. If logs/collectors is on a UNIX or Linux, system, provide the full path to the directory. If logs/collectors is on a Windows system, provide the virtual directory that points the logs/collectors folder. This variable is used by the CS_log_send script.
CS_PLATFORM	The SmartCloud Cost Management application server name.
CS_UPATH	The home directory for the account running the ituam_send_aacct script.
CS_METHOD=SFTP Or CS_METHOD=SCP	To use SFTP (Secure FTP, valid only if the SmartCloud Cost Management application server is on UNIX or Linux) or SCP (Secure Copy, valid only if SmartCloud Cost Management application server is on Windows), the user defined by the ITUAM_USER variable must have a null Secure Shell Public Key on the SmartCloud Cost Management application server for the account defined by ITUAM_USER. This enables the ITUAM_USER account to connect to SmartCloud Cost Management as the CS_USER account and no password is needed.
ITUAM_USER	The SmartCloud Cost Management account on the collector platform.
CS_USER	The receiving account on the SmartCloud Cost Management application server.
CS_KEY	Not applicable.
CS_COLL_PATH	The path to the logs/collectors directory on the SmartCloud Cost Management application server. If CS_METHOD=SCP, provide the full path to the logs/collectors folder. If CS_METHOD=SFTP, provide the full path from the login directory to the logs/collectors directory.
CS_PLATFORM	The SmartCloud Cost Management application server name.
CS_UPATH	Not applicable.

Using the ituam_send_aacct script

The ituam_send_aacct script is used to transport the usage logs to the SmartCloud Cost Management application server where they are processed and loaded into the SmartCloud Cost Management database.

Note: To use this script, uncomment the settings that call the script in the root account crontab file.

You can call the script with an optional date argument where date is in the format YYYYMMDD. The data argument specifies that only those usage logs that contain the specified date in the file name are transferred. If not argument is present, logs with the current data in the file name are transferred.

Virtual I/O Server metrics collected and SmartCloud Cost Management rate codes:

The tables in the following sections show the metrics that can be collected from the Virtual I/O data files by record type. The tables also show the SmartCloud Cost Management rate code that is assigned to each metric.

By default, only record type 10 metrics are collected as defined by the `AACCT_TRANS_IDS` variable in the `A_config.par` file . To collect other record types, the padmin user must modify the `A_config.par` file on the Virtual I/O Server.

Process Metrics Collected (Record Type 1)

Table 166. Advanced Accounting Process Metrics Collected

Metric	Rate Code
Interval count	**AAID0101**
Elapsed time in seconds	**AAID0102**
Elapsed thread time in seconds	**AAID0103**
CPU time in seconds	**AAID0104**
Elapsed disk pages in seconds	**AAID0105**
Elapsed real pages in seconds	**AAID0106**
Elapsed virtual memory pages in seconds	**AAID0107**
Local file I/O in MB	**AAID0108**
Other file I/O in MB	**AAID0109**
Local sockets I/O in MB	**AAID0110**
Other sockets I/O in MB	**AAID0111**

System Metrics Collected (Record Type 4)

Table 167. Advanced Accounting System Metrics Collected

Metric	Rate Code
Number of CPUs (interval aggregate)	**AAID0401**
Entitled capacity (interval aggregate)	**AAID0402**
System pad length (interval aggregate)	**AAID0403**
System idle time in seconds	**AAID0404**
User process time in seconds	**AAID0405**
Interrupt time in seconds	**AAID0406**
Memory size in MB (interval aggregate)	**AAID0407**
Large page pool in MB	**AAID0408**
Large page pool in-use in MB	**AAID0409**

Table 167. Advanced Accounting System Metrics Collected (continued)

Metric	Rate Code
Pages in	**AAID0410**
Pages out	**AAID0411**
Number start I/O	**AAID0412**
Number page steals	**AAID0413**
I/O wait time in 1/1000 seconds	**AAID0414**[1]
Kernel process time in 1/1000 seconds	**AAID0415**[1]
Interval elapsed time in 1/1000 seconds	**AAID0416**[1]

[1]These rate codes are not pre-loaded in the Rate table. To use these rate codes, you can load the codes into the Rate table using Administration Console. To convert these metrics from 1/1000 seconds to seconds, an Integrator stage can be added to the job file used to process type 4 records as shown in the following example:

<Stage name="ResourceConversion" active="true">

<Resources>

<Resource name="AAID0415">

<FromResources>

<FromResource name="AAID0415" symbol="a"/>

</FromResources>

</Resource>

</Resources>

<Parameters>

<Parameter formula="a/1000"/>

</Parameters>

</Stage>

File System Metrics Collected (Record Type 6)

Table 168. Advanced Accounting File System Metrics Collected

Metric	Rate Code
Bytes transferred in MB	**AAID0601**
Read/write requests	**AAID0602**
Number opens	**AAID0603**
Number creates	**AAID0604**
Number locks	**AAID0605**

Network Metrics Collected (Record Type 7)

Table 169. Advanced Accounting Network Metrics Collected

Metric	Rate Code
Network number I/O	**AAID0701**
Network bytes transferred in MB	**AAID0702**

Disk Metrics Collected (Record Type 8)

Table 170. Advanced Accounting Disk Metrics Collected

Metric	Rate Code
Transfers	**AAID0801**
Block reads	**AAID0802**
Block writes	**AAID0803**
Transfer block size (interval aggregate)	**AAID0804**

Virtual I/O Server Metrics Collected (Record Type 10)

Table 171. Advanced Accounting Virtual I/O Server Metrics Collected

Metric	Rate Code
Server bytes in	**AAID1001**
Server bytes out	**AAID1002**

Schedule the data collection and consolidation scripts

The scripts described in this section must be scheduled to run on a regular basis. You can use any batch scheduler to run these scripts; however, the scripts must be run under the root user account. During the SmartCloud Cost Management installation, the file `$ITUAM_UC_HOME/etc/cron.entry` was created. This file contains sample crontab entries for these scripts.

Note: If you installed the operating system or file system collectors remotely, the scripts described in this topic are scheduled automatically.

- **`$ITUAM_UC_HOME/etc/ituam_uc_nightly.`** This nightly collection script should be scheduled to run nightly around 1 a.m. If you use the example entry in the `cron.entry` file, output from this script is redirected to the log file `$ITUAM_UC_HOME/log/ituam_uc_nightly.log`.

  This script collects the raw UNIX and SmartCloud Cost Management accounting files and formats and sorts the files into one nightly accounting file. The script also executes the Sampler utility (`$ITUAM_UC_HOME/bin/A_sampler`) to get a snapshot of file system use. This snapshot is written to a nightly storage file.

  The nightly accounting and storage files are transferred to the `$ITUAM_UC_HOME/accounting/<nodename>` directory.
- **`$ITUAM_UC_HOME/etc/check_pacct.`** This script should be called three times each hour. It is used to manage the size of the UNIX/Linux process accounting (`pacct`) file. This file usually resides on the root file system in `/var/adm`. The location varies for different UNIX/Linux types.

  This script checks the size of the current `pacct` file. If the file has reached a threshold size (2000 blocks by default), the file is moved to `$ITUAM_UC_HOME/history/pacct_hold` and a new file is started.

- **$ITUAM_UC_HOME/scripts/enterprise/CS_nightly_consolidation.** This script should be scheduled to run nightly after the ituam_uc_nightly script has run. The CS_nightly_consolidation script consolidates the nightly accounting and storage files for the previous day. Refer to the comments in the beginning of the script to determine the best script configuration for your site.

  The CS_nightly_consolidation script produces CSR files, which are used as input to SmartCloud Cost Management.
- **$ITUAM_UC_HOME/scripts/enterprise/CS_send.** This script places the CSR files produced by the CS_nightly_consolidation script in a designated process definition directory on the SmartCloud Cost Management application server. The CS_send script should be run after the CS_nightly_consolidation script has run. Refer to the comments in the beginning of the script to determine the best script configuration for your site.

Collecting data: setting up the data collection scripts

During the data collection process, SmartCloud Cost Management Data Collectors gather data from files produced by the UNIX/Linux operating system and from the optional Oracle or DB2 accounting files that are produced by SmartCloud Cost Management.

The data collectors format the individual UNIX/Linux and SmartCloud Cost Management accounting files and produce one nightly accounting file. The records in the nightly accounting file contain a jobtype that specifies the source of the data in that record. The job types are software package; UNIX/Linux interactive, background, and storage; Oracle; Oracle table storage; DB2; and DB2 table storage.

The data collectors also optionally gather data from file systems and produce a nightly storage file. The records in the nightly storage file contain the jobtype file system.

The following scripts are used during the data collection process. All scripts are in the $ITUAM_UC_HOME/etc directory with the exception of the get_odb_storage and get_db2_storage scripts. These database storage scripts are in the $ITUAM_UC_HOME/scripts/oracle and db2 directories, respectively.

Note: The get_odb_storage and get_db2_storage scripts support the Oracle and DB2 database collectors.

- **Check pacct File script (check_pacct**). This script should be called three times each hour. It is used to manage the size of the UNIX pacct file. You might need to run this script more often; however, it is recommended that you do not run this script at the start of the hour because other jobs are usually scheduled at that time.
- **Nightly Accounting script (ituam_uc_nightly).** This script handles all of the steps in the data collection process, including calling the following scripts used for data collection. This script should be run only once a day. The variables required for this script and the following scripts are provided set in the A_config.par file.

  Note: You can use any batch scheduler to run the **check_pacct** and **ituam_uc_nightly** scripts; however, the scripts must be run under the root user account. During the installation of the collectors, the file **$ITUAM_UC_HOME/etc/cron.entry** was created. This file contains example crontab entries for these scripts.

- **Turn Accounting script (turnacct).** This script moves the raw UNIX and SmartCloud Cost Management accounting files into the `history` directory and prepares the files to be formatted by the `runacct` script.
- **Run Accounting script (runacct).** This script processes the raw UNIX and SmartCloud Cost Management accounting files and generates the nightly accounting file.
- **Sampler script (sampler).** This script calls the `A_sampler` utility to gather data from the Linux or Linux file system and produce the nightly storage file.
- **Database storage scripts (`get_odb_storage` and `get_db2_storage`).** These scripts collect Oracle data objects and datafile storage information and DB2 partition storage information, respectively.

Check pacct File Script (check_pacct)

The `check_pacct` script checks the size of the UNIX/Linux `pacct` file. You schedule this script to perform periodic checks through the clock daemon (cron).

The UNIX/Linux system imposes a limit on the size of the `pacct` file. When the usage of the root file system reaches 98 percent, the UNIX/Linux kernel will turn process accounting off without notification. No information is recorded until the usage of the root file system is below the system threshold.

The `check_pacct` script helps to assure that no UNIX/Linux accounting data is lost because the script maintains the size of the `pacct` below a user specified limit. When the `pacct` file exceeds the specified limit, this script places the current `pacct` file in a holding area in the `history` directory and re-initializes the live `pacct` file. Depending on the activity on your system, multiple `pacct` files can be generated in one day.

The `runacct` script processes the multiple `pacct` files and incorporates all the data in the nightly accounting file.

Nightly Accounting Script (ituam_uc_nightly)

This script handles all of the steps in the data collection process, including calling the following scripts used for data collection. This script should be run only once a day. The variables required for this script and the following scripts are provided set in the `A_config.par` file.

The Nightly Accounting script (`ituam_uc_nightly`) performs the following functions:

- Calls the `turnacct` script to move the raw UNIX and SmartCloud Cost Management accounting files to the `history` directory.
- Calls the `runacct` script to processes the raw UNIX and SmartCloud Cost Management accounting files and generate the nightly accounting file.
- If the `ITUAM_SAMPLE` variable is set to `Y` in the `A_config.par` file, calls the `sampler` script to get a snapshot of file system use and generate the nightly file system storage file.
- If the `ORACLE_STR_SAMPLE` variable is set to `Y` in the `A_config.par` file, calls the `get_odb_storage` script to collect table and datafile information and create the nightly Oracle storage file.
- If the `DB2_STR_SAMPLE` variable is set to `Y` in the `A_config.par` file, calls the `get_db2_storage` script to collect database partition storage information and create the nightly DB2 storage file.

Note: The get_odb_storage and get_db2_storage scripts support the Oracle and DB2 database collectors.

The ituam_uc_nightly script should be scheduled to run nightly around 1 a.m. Nightly processing keeps the accounting and storage data uniform and in sync for more precise and controllable resource management and chargeback.

You can use any batch scheduler to run this script; however, the script must be run under the root user account.

If you use the example entry in the $ITUAM_UC_HOME/etc/cron.entry file to run this script, output from this script is redirected to the log file $ITUAM_UC_HOME/log/ituam_uc_nightly.log.

Turn Accounting Script (turnacct)

The turnacct script moves the raw UNIX and SmartCloud Cost Management accounting files into the history directory and prepares the files to be formatted by the runacct script.

The turnacct script moves the following raw UNIX and SmartCloud Cost Management accounting files into the history directory and prepares the files to be formatted by the runacct script:

- UNIX/Linux wtmp, wtmpx, and pacct files.
- SmartCloud Cost Management Activity file ($ITUAM_UC_HOME/data/A_activity.sys) (optional). This file is created only if project accounting is enabled.
- SmartCloud Cost Management Oracle Accounting file ($ITUAM_UC_HOME/data/A_dbacct.sys) (optional). To collect data from this file, you must have the variable A_ORACLE_ACCT set to Y in the A_config.par file.
- SmartCloud Cost Management DB2 Accounting file ($ITUAM_UC_HOME/data/A_db2acct.sys) (optional). To collect data from this file, you must have the variable A_DB2_ACCT set to Y in the A_config.par file.

Note: The A_dbacct.sys and A_db2acct.sys files are used by the Oracle and DB2 database data collectors.

The turnacct script renames the file to include the date in YYYYMMDD format and then re-initializes the file to collect the next day's data.

The turnacct script is also used to turn UNIX/Linux process accounting on or off. For this use of the script, the command line arguments are:

```
turnacct on
turnacct off
```

Run Account Script (runacct)

Therunacct script processes the raw UNIX and SmartCloud Cost Management accounting files and generates the nightly accounting file.

Because the individual UNIX and SmartCloud Cost Management accounting files contain raw data in binary format, they must be formatted on the computer on which the data was collected. The runacct script runs the A_format utility against the raw UNIX and SmartCloud Cost Management accounting files in the history directory and creates temporary accounting files.

The runacct script chronologically sorts the records in the temporary accounting files and produces one nightly accounting file, acc_<date>.dat, where <date> is in YYYYMMDD format.

This runacct script uses the first sixteen characters of the statistics record to sort. These characters contain the date and record type, which are unique, to assure proper sorting. Under UNIX, the UNIX sort command that is used is: sort +0.0 -0.16.

If your temporary accounting files are very large and the TEMP directory is too small for sorting, you might need to add the option -T to the UNIX sort command in the runacct script.

The following is a sample of the CLI commands for the A_format utility to format accounting data from within the runacct script:

```
# A_format
 FORMAT> FORMAT/TYPE=ACTIVITY/ZERO "/usr/ituam/history/activity_<date>"
 %FRMT-I-PROCFILE, processing file-"usr/ituam/historyactivity_<date>
 FORMAT> FORMAT/TYPE=ACCT/ROLL/ZERO "/usr/ituam/history/pacct_<date>"
 %FRMT-I-PROCFILE, processing file-"usr/ituam/history/pacct_<date>"
 FORMAT> FORMAT/TYPE=WTMP/ZERO "/usr/ituam/history/wtmp_<date>"
 %FRMT-I-PROCFILE, processing file-"usr/ituam/history/wtmp_<date>
 FORMAT> EXIT
 #
```

The /ZERO qualifier re-initializes the temporary accounting file. This qualifier ensures that only the current day's statistics are contained in the file.

Using the /ROLL Qualifier

The A_format utility has an additional qualifier when processing the UNIX image accounting data in the UNIX pacct file. This qualifier is /ROLL. The /ROLL qualifier combines similar image records from background jobs into a rolled accounting record. When the same image is run by a user several times in background (no controlling terminal), these image records are rolled into one roll-up record that indicates the number of times the image was executed. This qualifier can reduce the size of the resulting nightly accounting file when numerous background jobs are performed.

Using the /AGGREGATE Qualifier

The /AGGREGATE qualifier instructs the A_format utility to treat all process records as background jobs. This means that all process records are rolled up based on username and processname. This qualifier has meaning only when used with the /ROLL qualifier.

Sampler Script (sampler)

The sampler script invokes the A_sampler utility. The A_sampler utility traverses the file systems that include specific directory trees that are defined in the Storage Parameter file ($ITUAM_UC_HOME/data/A_storage.par).

To prevent double counting of NFS and automounted file systems, the A_sampler utility traverses only locally mounted file systems.

The A_sampler utility writes the sampled disk space usage information to the Storage file ($ITUAM_UC_HOME/data/A_storage.sys). The A_sampler utility accumulates the amount of file space allocated to the file system and the amount of file space used by UID and GID.

The sampler script moves the A_storage.sys file to the history directory and renames the file str_<date>.dat, where <date> is in YYYYMMDD format.

To execute the sampler script, the variable ITUAM_SAMPLE must be set to Y in the $ITUAM_UC_HOME/data/A_config.par file.

Database Storage Scripts (get_odb_storage and get_db2_storage)

The get_odb_storage and get_db2_storage scripts collect Oracle data objects and datafile storage information and DB2 partition storage information, respectively.

Note: The get_odb_storage and get_db2_storage scripts are supported by the Oracle and DB2 database collectors.

If the ORACLE_STR_SAMPLE variable is set to Y in theA_config.par file, the ituam_uc_nightly script calls the get_odb_storage script to collect Oracle data objects and datafile storage information. This information is output into the $ITUAM_UC_HOME/history/ora_sto_<date>.dat file.

If the DB2_STR_SAMPLE variable is set to Y in theA_config.par file, the ituam_uc_nightly script calls the get_db2_storage script to collect DB2 database partition storage information. This information is output into the $ITUAM_UC_HOME/history/db2_sto_<date>.dat file.

Redo Nightly Script (redo_nightly)

Use the redo_nightly script to reprocess files if the ituam_uc_nightly script fails.

If the ituam_uc_nightly script fails, use this script to reprocess the following files in the history directory:

- pacct_<date>
- wtmp_<date>
- activty_<date>
- dbacct_<date>
- db2acct_<date>

The redo_nightly script accepts the following command line arguments:

```
redo_nightly  today  yesterday  all  yyyymmdd
```

Where:

- today=process files with the current day's date (this is the default)
- yesterday=process files with yesterday's date (the day before the current day)
- all=process files for all dates
- yyyymmdd=process files with a specific date (the date must be in yyyymmdd format)

The redo_nightly script invokes the runacct script, which creates and sends a new nightly accounting file or files to the $ITUAM_UC_HOME/accounting/<nodename> directory. Therefore, existing nightly accounting files for the same date are overwritten. If you want to save existing files, you must move them out of the

accounting/<nodename> directory.

Consolidating data: setting up the data consolidation scripts

During the data consolidation process, the data collectors process the nightly accounting and storage files that were created by the data collection scripts and produce an output CSR file. SmartCloud Cost Management processes the data in the CSR file and provides comprehensive job accounting, chargeback, and cost analysis capabilities in addition to capacity and resource reporting.

The following scripts are used during the data consolidation process.

- **Nightly Consolidation script ($ITUAM_UC_HOME/scripts/enterprise/CS_nightly_consolidation)**. This script calls the following scripts that support the consolidation process. The variables required for this script and the following scripts are set in the A_config.par file.
  - **Process Multiple Nodes script ($ITUAM_UC_HOME/etc/proc_multi).** This script consolidates the nightly accounting and storage files.
  - **CS Generate Summary script ($ITUAM_UC_HOME/scripts/enterprise/CS_gen_sum).** This script generates CSR files from the consolidated data produced by the proc_multi script.

Nightly Consolidation Script (CS_nightly_consolidation)

The CS_nightly_consolidation script should be scheduled to run nightly after the ituam_uc_nightly script has been run. Refer to the comments in the beginning of the script to determine the best script configuration for your site.

The CS_nightly_consolidation script does the following:

- Calls the proc_multi script to consolidate the nightly accounting and file system storage files for the previous day.
- Calls the CS_gen_sum file to generate CSR files from the consolidated accounting and storage files.
- Concatenates the nightly Oracle and DB2 storage files for the previous day to create CSR files of tablespace and datafile usage.
- Calls the CS_fs_resource utility to read nightly storage files and generate CSR file of file system usage.
- Places the CSR files in the $ITUAM_UC_HOME/CS_input_source directory. The files are transferred from this directory to the SmartCloud Cost Management application server.

Running a UNIX script file from within a job file

It is possible to run a UNIX script file from within their Job Runner job file. This section describes how that can be done. A sample job file and script file have been added to the Tivoli SmartCloud Cost Management samples.

Beginning the shell script

The script file that Job Runner calls should start with this line:

```
#!/bin/sh
```

If Job Runner runs a script that does not start with that line, Job Runner will most likely fail with a “No step output” error.

The “No step output” error occurs because a script that does not start with #!/bin/sh inherits environment variable values from SmartCloud Cost

Management's Embedded WebSphere (eWAS) environment rather than the environment variable values that are seen in a typical UNIX terminal session. For example, the SHELL and SHELLOPTS environment variable values look similar to the following for a sample Red Hat Enterprise Linux 5 terminal session:

```
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor
```

On the other hand, the SHELL and SHELLOPTS environment variable values look similar to this for SmartCloud Cost Management's eWAS environment:

```
SHELL=com.ibm.ws.management.tools.WsServerLauncher
SHELLOPTS=braceexpand:hashall:interactive-comments:posix
```

Shell script must return zero

Your shell script must return a zero return code to Job Runner. If your shell script returns a non-zero return code, Job Runner will stop processing the job file that called the script and log an error.

An example of a UNIX command that routinely returns a non-zero return code even when processing is successful is:

```
/bin/netstat -V
```

In this case, netstat returns a return code of 5. On Linux, you can see this behavior by first running `/bin/netstat -V` from a terminal session and then run this command:

```
echo $?
```

`$?` contains the return status of the last command. Since the `netstat` command is the last (and only) command executed on the command line, its return value (5) is echoed.

If your shell script can return a non-zero return code, even when it is successful, you should add processing to your script that returns a zero (success) status to Job Runner. There are any number of ways to do this, but here are two examples:

- Catch a particular return code (in this example, the rc is 5) and reset it to 0.

  ```
  if test $? = "5"
  then
      exit 0
  fi
  ```

- Unconditionally set the "last command executed" return code to 0 by adding this line to the end of the script:

  ```
  ls > /dev/null
  ```

  This command will always set a success status of 0.

Example

Here is an example of a very simple script that meets the "run a script in Job Runner" criteria:

```
#!/bin/sh
/bin/netstat -V
if test $? = "5"
then
exit 0
fi
```

Sample files

A sample job file, SameRunScript.xml, which is contained in <SCCM_install_dir>/samples/jobfiles, runs the sample script named run_netstat_sample.sh, which is located in the <SCCM_install_dir>/bin directory of a Linux install.

When you create your own script file, you can place it in a directory of your choice. If you do not include the fully qualified path for your script file, Job Runner searches its <SCCM_install_dir>/collectors, <SCCM_install_dir>/bin, and <SCCM_install_dir>/lib directories, in that order, for the script.

When Job Runner runs the run_netstat_sample.sh script, information similar to the following is logged in the job log.

```
      ** Begin Step Output **
      net-tools 1.60
      netstat 1.42 (2001-04-15)
      Fred Baumgarten, Alan Cox, Bernd Eckenfels, Phil Blundell, Tuan Hoang and others
      +NEW_ADDRT +RTF_IRTT +RTF_REJECT +FW_MASQUERADE +I18N
      AF: (inet) +UNIX +INET +INET6 +IPX +AX25 +NETROM +X25 +ATALK +ECONET +ROSE
      HW: +ETHER +ARC +SLIP +PPP +TUNNEL +TR +AX25 +NETROM +X25 +FR +ROSE +ASH +SIT +FDDI
  +HIPPI +HDLC/LAPB
      **End Step Output **
```

CSR file types

The data consolidation produces one or more CSR file.

The data consolidation produces one or more of the CSR files shown in the following table. Although the format of each CSR file type is the same, the data that appears in the files depends on the environment variables that you set in the A_config.par file and the parameters that you pass from the CS_nightly_consolidation script.

Table 172. CSR Files Produced by the CS_gen_sum Script

CSR File	Description
CS_sum_<date>.csv	By default, the CS_nightly_consolidation script passes the following parameters to the CS_gen_sum script to produce this CSR file: package node user packagename The package parameter specifies that UNIX/Linux package/process jobtype is included in the CSR file. The keywords node, user, and packagename specify that the identifiers SYSTEM_ID, USERNAME, and PROCESSNAME appear in the CSR records.
CS_sum_ora_<date>.csv	To produce this CSR file containing Oracle metrics, the environment variable GEN_ORACLE must be set to Y in the A_config.par file. By default, the CS_nightly_consolidation script passes the following parameters to the CS_gen_sum script to produce this CSR file: oracle node user or_base or_user The oracle parameter specifies that the Oracle jobtype is included in the CSR file. The keywords node, user, or_base, and or_user specify that the identifiers SYSTEM_ID, USERNAME, OR_BASE, and OR_USER appear in the CSR records.

Table 172. CSR Files Produced by the CS_gen_sum Script (continued)

CSR File	Description
CS_sum_orasto_<date>.csv	To produce this CSR file containing Oracle table storage metrics, the environment variable GEN_ORACLE_STORAGE must be set to Y in the A_config.par file. This file is created by the CS_nightly_ consolidation script, which concatenates the individual Oracle storage files.
CS_sum_db2_<date>.csv	To produce this CSR file containing DB2 metrics, the environment variable GEN_DB2 must be set to Y in the A_config.par file. By default, the CS_nightly_consolidation script passes the following parameters to the CS_gen_sum script to produce this CSR file: db2 node user db2_base The db2 parameter specifies that the DB2 jobtype is included in the CSR file. The keywords node, user, and db2_base specify that the identifiers SYSTEM_ID, USERNAME, and DB2_BASE appear in the CSR records.
CS_sum_db2sto_<date>.csv	To produce this CSR file containing DB2 table storage metrics, the environment variable GEN_DB2_STORAGE must be set to Y in the A_config.par file. This file is created by the CS_nightly_ consolidation script, which concatenates the individual DB2 storage files.
CS_sum_fs_<date>.csv	To produce this CSR file containing UNIX/Linux file system metrics, the environment variable GEN_UNIXFS must be set to Y in the A_config.par file (this is the default). The CS_nightly_consolidation script calls the CS_fs_resource utility to read the nightly storage files and generate the records for this CSR file.

Transferring CSR files to the SmartCloud Cost Management application server

The CS Send script ($ITUAM_UC_HOME/scripts/enterprise/CS_send) transfers the CSR files produced during the consolidation process from the CS_input_source directory to one of the process definition directories on the SmartCloud Cost Management application server. The destination process definition directory depends on the CSR file type as shown in the following table.

Each process definition directory contains feed subdirectories that reflect the source of the data. For example, UNIX file system CSR files (CS_sum_fs_<date>.csv) from the server zeus are placed in the subdirectory zeus within the process definition directory UnixFS (for example, UnixFS/zeus) on the SmartCloud Cost Management application server. If the subdirectory does not exist in the process definition directory, it is created the first time that the CS_send script is run.

Note: The **CS_send** script also renames the CSR files when the files are transferred to a feed subdirectory in the process definition folders. For example, all CS_sum_fs_<date>.csv CSR files are sent to a subdirectory in the UnixFS process definition directory and are renamed **<date>.txt.**

Table 173. Process Definition Folders and CSR Files

Destination Process Definition Folder Name	CSR File Type
processes/UnixDB2	The CS_sum_db2_<date>.csv CSR files are sent to a feed subdirectory within this directory.
processes/UnixDB2Storage	The CS_sum_db2_stor_<date>.csv CSR files are sent to a feed subdirectory within this directory.
processes/UnixORA	The CS_sum_ora_<date>.csv CSR files are sent to a feed subdirectory within this directory.
processes/UnixORAStorage	The CS_sum_ora_stor_<date>.csv CSR files are sent to a feed subdirectory within this directory.
processes/UnixFS	The CS_sum_fs_<date>.csv CSR files are sent to a feed subdirectory within this directory.
processes/UnixOS	The CS_sum_<date>.csv CSR files are sent to a feed subdirectory within this directory.

The CS_send script should be run after the CS_nightly_consolidation script has completed. Refer to the comments in the beginning of the script to determine the best script configuration for your site.

Setting the configuration variables for file transfer

To transfer the CSR files to the SmartCloud Cost Management application server using the CS_send script, set the following variables in the $ITUAM__UC_HOME/data/A_config.par file.

The variable descriptions are grouped by CS_METHOD variable value because this value determines the values that you can set for other variables. The CS_METHOD variable defines the protocol that is used to transfer the CSR files to the SmartCloud Cost Management application server.

Table 174. Variables for File Transfer to the SmartCloud Cost Management Application Server

Variable	Description
CS_METHOD=MV	Move.
CS_USER	Not applicable.
CS_KEY	Not applicable.
CS_PROC_PATH	The path to the processes directory on the SmartCloud Cost Management application server. If the application server is on a Windows system, this variable is not applicable. If the application server is on a UNIX or Linux system, provide the full path to the procesess directory. This variable is used by the CS_send script.

Table 174. Variables for File Transfer to the SmartCloud Cost Management Application Server (continued)

Variable	Description
CS_COLL_PATH	The path to the log/collectors directory on the SmartCloud Cost Management application server. If the application server is on a Windows system, this variable is not applicable. If the application server is on a UNIX or Linux system, provide the full path to the log/collectors directory.
CS_PLATFORM	Not applicable.
CS_UPATH	Not applicable
CS_METHOD=FTP	
CS_USER	The account required to log on to the SmartCloud Cost Management application server. If the application server is on a Windows system and you are using a domain account, use the following format for the user name: CS_USER="<domain name>\\\\<account name>" Make sure that you use the four backslashes between the account name and domain name as shown.
CS_KEY	The password for the account defined by the CS_USER variable.
CS_PROC_PATH	The path to the processes directory on the SmartCloud Cost Management application server. If processes is on a UNIX or Linux, system, provide the full path to the directory. If processes is on a Windows system, provide the virtual directory that points the processes folder. This variable is used by the CS_send script.
CS_COLL_PATH	The path to the logs/collectors directory on the SmartCloud Cost Management application server. If logs/collectors is on a UNIX or Linux, system, provide the full path to the directory. If logs/collectors is on a Windows system, provide the virtual directory that points the logs/collectors directory. This variable is used by the CS_log_send script.
CS_PLATFORM	The SmartCloud Cost Management application server name.
CS_UPATH	The home directory for the account running the CS_send or CS_log_send script.
CS_METHOD=SFTP Or CS_METHOD=SCP	To use SFTP (Secure FTP, valid only if the SmartCloud Cost Management application server is on UNIX or Linux) or SCP (Secure Copy, valid only if the SmartCloud Cost Management application server is on Windows), the user defined by the ITUAM_USER variable must have a null Secure Shell Public Key on the SmartCloud Cost Management application server for the account defined by ITUAM_USER. This enables the ITUAM_USER account to connect to the SmartCloud Cost Management application server as the CS_USER account and no password is needed.

Table 174. Variables for File Transfer to the SmartCloud Cost Management Application Server (continued)

Variable	Description
ITUAM_USER	The SmartCloud Cost Management account on the collector platform.
CS_USER	The receiving account on the SmartCloud Cost Management application server.
CS_KEY	Not applicable.
CS_PROC_PATH	The path to the procesess directory on the SmartCloud Cost Management application server. If CS_METHOD=SCP, provide the full path to the Processes folder. If CS_METHOD=SFTP, provide the full path from the login directory to the processes directory. This variable is used by the CS_send script.
CS_COLL_PATH	The path to the logs/collectors directory on the SmartCloud Cost Management application server. If CS_METHOD=SCP, provide the full path to the logs/collectors directory If CS_METHOD=SFTP, provide the full path from the login directory to the logs/collectors directory. This variable is used by the CS_log_send script.
CS_PLATFORM	The SmartCloud Cost Management application server name.
CS_UPATH	Not applicable.

Transferring log files to the SmartCloud Cost Management application server

You can send UNIX/Linux subsystem log files that contain usage data directly to the SmartCloud Cost Management application server for conversion to a CSR file and subsequent processing. For example, you might want to transfer the xferlog produced by some ftpd systems directly to the SmartCloud Cost Management application server.

You can use the CS Log Send script ($ITUAM_UC_HOME/scripts/enterprise/CS_log_send) to transfer the log files to the SmartCloud Cost Management application server. The log file is placed in the logs/collectors directory on the application server in the following subdirectory architecture: logs/collectors/<collector>/<feed>.

Note: To transfer log files to the SmartCloud Cost Management application server using the CS_log_send script, you must set the same variables in the A_config.par file.

Chapter 8. Troubleshooting

Use this topic to find troubleshooting information for common issues.

Troubleshooting information

In addition to the troubleshooting information provided here, you can also refer to technotes to obtain late-breaking information about problems, limitations, and workarounds.

Common problems and solutions

This section shows you how to solve typical problems that might arise when using IBM SmartCloud Cost Management.

Table 175. Problems and solutions for common problems

Problem	Potential solution
Issues when using the **Rate Tables** panel in Microsoft Internet Explorer	When using the new **Rate Tables** in Microsoft Internet Explorer, you may see caching issues where new items or changes are not immediately visible in the panel. The workaround is to change a setting in Internet Explorer. To do this: • Go to **Tools** > **Internet Options** > **Browsing History** > **Settings**. • Check for newer versions of stored pages, make sure "**Every time I visit the webpage**" is selected, rather than "**Automatically**".
When using the new **Rate Tables** panel, occasionally a message may be displayed saying `Sorry, an error has occurred`.	If you see this message, we recommend that you clear your cache and cookies from the server, then refresh the page. If this does not resolve the problem, logout and log back into the Administration Console.

Troubleshooting installation

Use this information if you are having problems with installation functions.

Manually uninstalling

A manual uninstall might be required in the unusual situation in which SmartCloud Cost Management does not uninstall cleanly.

The uninstall of SmartCloud Cost Management is straight forward but a manual uninstall may be preferred if you want to retain configuration files, job files or logs as these are all removed by a standard uninstall.

© Copyright IBM Corp. 2013, 2015

Manually uninstalling SmartCloud Cost Management:

Manually uninstalling SmartCloud Cost Management is a straightforward process. Simply decide what items you want to retain, then perform the steps below.

Procedure

1. Stop any running servers:

   ```
   <SCCM_install_dir>/bin/stopServer.sh mcs
   <SCCM_install_dir>/bin/stopServer.sh sccm
   ```

2. Back up any files you want to keep:

   ```
   SCCM logs - <SCCM_install_dir>/logs
   Server logs - <SCCM_install_dir>/wlp/usr/servers/<sccm or mcs>/logs
   Jobfiles - <SCCM_install_dir>/jobfiles
   Custom objects - <SCCM_install_dir>/setup/dbobjects/custom
   Configuration files - <SCCM_install_dir>/config
   ```

3. Remove SmartCloud Cost Management:

   ```
   rm -rf <SCCM_install_dir>
   ```

4. Optionally remove Java. If SmartCloud Cost Management was installed as root, Java 7 was installed as an RPM. This can be removed as follows:

   ```
   rpm -e ibm-java-x86_64-sdk-8.0
   ```

Manually uninstalling Windows Process Collector:

A manual uninstall might be required in the unusual situation in which SmartCloud Cost Management Windows Process Collector does not uninstall cleanly.

Procedure

1. Remove the Installshield registry directory. The following is an example directory:

   ```
   C:\Program Files\Common Files\InstallShield\Universal\IBM_TUAM_WPC
   ```

2. Rename the Tivoli common logs install directory for SmartCloud Cost Management. You should rename the directory rather than removing it in case logs are needed by IBM Support for additional problem determination. The following is an example directory:

   ```
   C:\Program Files\ibm\tivoli\common\AUC\logs\install
   ```

3. Remove the Windows Process Collector autostart service as follows:
   a. Download the Windows Server 2003 or 2008 Resource Kit..
   b. Stop the service named SmartCloud Cost Management Process Collector. To stop the service, click**Start** > **Control Panel** > **Administrative Tools** > **Services**. Right-click the service, and then click **Stop**.
   c. Run the Windows Resource kit command line utility `instsrv.exe` as shown in the following example. (An example path for the `instsrv.exe` utility is `C:\Program Files\Windows Resource Kits\Tools`).

      ```
      C:\temp>instsrv CIMSWinProcess REMOVE
      ```

      Note: To find the service name to use in the command, right-click the service, and then click **Properties**. The service name is the last value in the **Path to executable** box.

4. Remove the Windows Process Collector Windows registry keys `HKEY_LOCAL_MACHINE\SOFTWARE\CIMS`.
5. Remove directory in which SmartCloud Cost Management is installed. For example, `C:\Program Files\ibm\tuam`.

Unable to connect to the Administration Console

After installing SmartCloud Cost Management, you may encounter an issue with accessing the Administration Console.

Symptom

You are unable to access the Administration Console after the install due to a timeout or an `Unable to connect` error message in your web browser.

Cause

This may be due to the firewall configuration on the system.

Solution

Open the port as user root as follows:

```
iptables -I INPUT -p tcp --dport console_port -j ACCEPT
service iptables save
```

where **`console_port`** is the https port chosen during installation, or 9443 by default. You should now be able to access the Administration Console successfully.

Troubleshooting administration

Use this information if you are having problems with administrative functions.

Server instance ports in use

Following the installation of SmartCloud Cost Management, which includes SmartCloud Cost Management and the Metering Control Service (MCS), the server instances that are running are configured out-of-the-box for HTTP and HTTPS ports. Situations may arise where these default ports must be changed. A port in use would be an example of this: "`TCP Channel defaultHttpEndpoint initialization did not succeed. The socket bind did not succeed for host localhost and port 8080. The port might already be in use.`"

About this task

The SmartCloud Cost Management server instance default ports are as follows:

- HTTP Port of 9080
- HTTPS Port of 9443

The MCS server instance default ports are as follows:

- HTTP Port of 8080
- HTTPS Port of 8077

In relation to a particular SmartCloud Cost Management instance, the following steps must be completed to resolve the issue:

Procedure

1. If the SmartCloud Cost Management port(s) must be changed:
   - Open the following file in a text editor: `<SCCM_install_dir>/wlp/usr/servers/sccm/server.xml`
   - Update the following element: `<httpEndpoint host="*" httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint">`
   - The server instance must be restarted to enable the port changes.
2. If the MCS port(s) must be changed:

- Open the following file in a text editor: `<SCCM_install_dir>/wlp/usr/servers/mcs/server.xml`
- Update the following element: `<httpEndpoint host="localhost" httpPort="8080" httpsPort="8077" id="defaultHttpEndpoint"/>`
- The server instance must be restarted to enable the port changes.

Results

The server instances now start without any port conflicts.

Active Reports do not open or are incorrectly displayed when rendered in Japanese, Chinese, or Korean

Use this topic to resolve the issue of active reports, such as the Project Summary report, in Tivoli Common Reporting hanging when opened, or displaying as a page of text when the report is rendered in Japanese, Chinese or Korean.

Symptom:

On some versions of Windows, when running an active report in Tivoli Common Reporting, opening the report in a browser results in the report hanging and not opening. Alternatively, when opening the report, it can be displayed as a page of text.

Cause:

This is caused by a bug with the `mht` file format because some browsers cannot open a file, which has a name containing Unicode characters.

Solution:

Rename the file so it does not contain Unicode characters and open the report in the browser.

Administration Console runs slowly, hangs, or will not connect to the database

If you are experiencing problems with Administration Console (for example the application runs slowly, hangs, or will not connect to the database despite the fact that you have a working data source connection), first stop and restart the application server. If you cannot restart the application server or the problem continues, reset Administration Console.

Note: For more information about stopping and starting the application server, see the related concept *Stopping and staring the application server*.

Unable to run Tivoli Common Reporting reports that use account code prompts

Use this topic if you are unable to run Tivoli Common Reporting reports that use account code prompts.

Symptom:

The **Account Structure**, **Account Code Level**, **Starting Account Code** and **Ending Account Code** prompts are not populated and the **Finish** button is not selectable when a SmartCloud Cost Management report is run in Tivoli Common Reporting.

Cause:

This may happen if you are using a user, for example the default `smadmin` user that has not been created correctly in SmartCloud Cost Management to run Tivoli Common Reporting reports.

Solution:

It is not recommended to use the default administrator user `smadmin`. Instead, you should create a specific reporting user or users that are used to run specific Tivoli Common Reporting reports. To do this:

- Create the user in the Central User Registry. The user is then created in SmartCloud Cost Management as part of the scheduled OpenStack context process job or this process can be manually run to add the user immediately.

For more information about creating users in SmartCloud Cost Management, see the related topic on Security and the Context job in the Configuration and Administering data collectors guides.

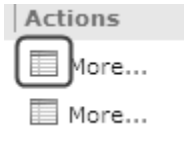
Report data not updated when running reports

When clicking the name of a report to run, the data in the report does not update and the same report is returned each time you run the report. This is expected behavior in Cognos, as the default value in the report properties is set to **View most recent report**.

About this task

Use this task to change the report properties to always run a report and display the latest data when the report name is clicked.

Procedure

1. Log in to your reporting interface.
2. Navigate to the **Common Reporting** package in the Public Folders view.
3. Open the **Common Reporting** package in the **Public Folders** view.
4. Navigate to the required report.
5. Click the 

Set Properties Menu icon.
6. In the **Report** tab, select **Run the Report** from the **Default action** drop-down list and press **OK**.
7. Alternatively, if you do not want to update the report properties, navigate to the required report and select the run Menu icon to run the report.

Results

When run, the report data is updated with the latest content.

Issues when using the SmartCloud Cost Management rate template

What should I do if there are issues when importing a rate template or labels on rate dimension fields are incorrect?

Symptom:

Issues may be seen if the offering XML is written in the incorrect structure. As a result of this, the following issues may occur:

- Rates are not displayed correctly in the rate tables in SmartCloud Cost Management.
- You may encounter issues when importing a rate template into a rate table.
- The labels on some of the rate dimension fields may be incorrect or undefined.

Solution:

Ensure that the offering XML is written in the correct structure. Use the sample offering XML that is shipped with the product. Its default location is in `<SCCM_install_dir>/offerings/offerings.xml`. For more information, see the *Administering the system* guide.

Troubleshooting database applications

Use this information if you are having problems with database functions.

Troubleshooting for Exception: DB2 SQL error: SQLCODE: -964, SQLSTATE: 57011, SQLERRMC: null

An error similar to the following can occur during database update operations: `Exception: DB2 SQL error: SQLCODE: -964, SQLSTATE: 57011, SQLERRMC: null`. This error can occur because the transaction log space is depleted or because of a temporary increase in the number of active transactions.

The following is a possible solution to this problem:

1. From the DB2 UDB Command Line Processor (CLP), run the following DB2 command:

   ```
   db2 get snapshot for all on sccm
   ```

2. Examine the values of the entries **Log space available to the database**; **Log space used by the database**; and **Secondary logs allocated currently**. These entries should indicate that the database is running low on available log space.
3. Increase the number of secondary log files available to the database by 12 to provide additional log file space. From the DB2 UDB CLP, run the DB2 command:

   ```
   db2 update db cfg for sccm using logsecond x
   ```

   Where `x` is the current value of secondary log space plus 12.

If the problem occurs again, the transaction log space shortage could be caused by in-doubt transactions in DB2. In-doubt transactions can be caused by prior server failures or crashes, which cause the transaction log to become full when transactions are performed. From the DB2 UDB CLP, connect to the SmartCloud Cost Management database, then run the following command:

```
db2 list indoubt transactions with prompting
```

Roll back any transactions with a timestamp near the time of the server crash.

Related problems

If your environment does not have enough memory or hard disk space, a message similar to the following can be generated:

```
AUCPE0202E The DBLoad process completed unsuccessfully with the following exception:
com.ibm.db2.jcc.b.SqlException: Error for batch element #306: DB2 SQL error:
SQLCODE: -964, SQLSTATE: 57011, SQLERRMC: null.
```

Solutions:

- Run SmartCloud Cost Management on an environment that has more memory and hard disk space.
- If you are using a DB2 database, use the preceding steps to increase the number of secondary log files.

Stored Procedure performance issues on DB2

There is a script called **DB2Utility** in IBM SmartCloud Cost Management 2.1.0.5, that allows for certain DB2® utilities to be called from the SmartCloud Cost Management installation. This script has various options, these include getting and setting reopt settings on Stored procedures and running the DB2 RUNSTATS command on SmartCloud Cost Management tables.

This script has the following options:

Table 176. DB2Utility.sh -h

Options	Descriptions
`-help, -h`	Print this message.
`-datasource, -d`	The DB2 datasource which should be used (as defined in the ISC), if none is specified the Admin datasource will be used.
`-runstats`	Runstats with default options for all tables.
`-runstats <table name> <options>`	Runstats where a single table can be specified and runstats options to be run on that table E.g -runstats CIMSSUMMARY AND INDEXES ALL.
`-getreopt`	Display the reopt settings for all SmartCloud Cost Management Stored Procedures.
`-getreopt <stored procedure>`	Display the reopt settings for the SmartCloud Cost Management Stored Procedures passed as a parameter.
`-reopt <stored procedure> <reopt option>`	Runs the **reopt** command on the Stored Procedure specified with the reopt option: i.e. NONE, ONCE or ALWAYS for example -reopt **GET_SUMMARY ONCE**.
`-reopt -f <config file>`	File containing reopt config for Stored Procedures for example: -reopt -f reopt_settings.config. The format in the file should be as follows: • GET_SUMMARY,ONCE • GET_SUMMARY_DAY,NONE

This script is located in: `<SCCM_install_dir>/bin`.

Note: Only DB2 9.7 versions onwards are supported by this script.

Using log files

SmartCloud Cost Management provides a variety of log files that provide informational and troubleshooting data. If you require assistance from IBM Software Support, you might be asked to provide one or more of these files.

Message and trace log files

Message and trace log files provide results for SmartCloud Cost Management .

The following message and trace log files are in the *<SCCM_install_dir>*/logs/server directory. In general, trace files are most useful for technical support while message files provide more user-friendly information that is translated into multiple languages.

- **trace<*archive index number*>.log.<*process index number*>**

  This log file contains trace messages generated by the SmartCloud Cost Management application.

- **message<*archive index number*>.log.<*process index number*>**

  This log file contains information, error, and warning messages generated by the SmartCloud Cost Management application.

The archive index number specifies the chronology of the archived files. The higher the archive index number, the older the file. Therefore, zero (0) is the current file. All files from one (1) on are archived files. You set the number of archived files that you want to retain in the logging.properties file. For more information about this file, see the related topic about setting logging options.. The default is 11.

The process index number is used by the Java Virtual Machine (JVM). Multiple virtual machines running concurrently cannot access the same trace or log files. Therefore, separate trace and log files are created for each virtual machine. The process index number is not configurable.

Configuring message and trace logs

You can set the maximum log file size, the level of detail that you want provided in the files, and the number of files that you want to archive in the <SCCM_install_dir>/config/logging.properties file. After making the changes in this file, the application server must be restarted.

Job log files

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 or failure occurred. You can view job log files in Administration Console.

Note: The SmartCloud Cost Management trace and message log files in the *<SCCM_install_dir>*/logs/server directory contain additional details about the jobs that are run.

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 Administration Console before scheduling the job to run in batch.

Setting the job log file directory path

The path to the job log files is defined in the `config.properties` file. For more information about this file, see the related topic about setting processing options.

Defining the log file output type

You can produce log data in a text file, an XML file, or both. By default, both a text file and an XML file are produced.

If want to produce one file type only, include the attribute `joblogWriteToTextFile="false"` or `joblogWriteToXMLFile="false"` in the job file.

If you want to view the log files in Administration Console, the job log files must be in XML format.

Text log files are named `yyyyMMdd_hhMMss.txt`. XML log files are named `yyyyMMdd_hhMMss.xml`. Where `yyyyMMdd_hhMMss` is the job execution time

Sending log files through email

You can choose to have output log files sent using email to a recipient or recipients. To send log files via email, set the appropriate SMTP attributes in the job file.

The log files that are attached to an email message are `.txt` files, regardless of the value for the attribute `joblogWriteToTextFile`. If both the `joblogWriteToTextFile` and `joblogWriteToXMLFile` attributes are set to `"false"`, the email message is empty and no log file is attached.

Note: If the `smtpSendJobLog` is set to `"true"`, and the email message cannot be delivered, a warning message is logged in the SmartCloud Cost Management message and trace files and in the command console.

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.

Embedded Web Application Server log files

WebSphere Liberty log files provide results for various functions related to SmartCloud Cost Management.

These log files are in the `<SCCM_install_dir>/wlp/usr/servers/sccm/logs` directory. The most important logs in this directory are:

`console.log`
: This log captures the application server system messages and **`System.out`** output generated by the SmartCloud Cost Management application.

trace.log
This WebSphere JVM log captures trace output generated by the application server.

Installation log files

A log file is created each time that SmartCloud Cost Management, SmartCloud Cost Management Windows Process Collector, or DB2 Runtime Client are installed or uninstalled. This log file provides results for each step in the install or uninstall process. If a warning or failure occurs during the installation or unistallation, the file indicates at which point the warning or failure occurred.

The install log files for SmartCloud Cost Management Windows Process Collector are stored in:

```
%PROGRAMFILES%\IBM\tivoli\common\AUC\logs\install
```

The install log files for SmartCloud Cost Management are stored in:

```
<SCCM_install_dir>/logs/install.log
```

Getting SmartCloud Cost Management system information

Use the `sccmProductInfo.sh` script to get information about the SmartCloud Cost Management installation. This information includes the SmartCloud Cost Management version that is installed, the data collectors that are installed, and information about the database or databases used by SmartCloud Cost Management.

About this task

Running the `sccmProductInfo` script returns useful SmartCloud Cost Management configuration information including the following:

- Dynamic class loading verification
- Data source verification
- Database initialization verification
- Database schema validation
- Database driver validation

To run the `sccmProductInfo` script from the command line:

- **Linux:** Using a shell command, type the following. Insert the path to your SmartCloud Cost Management installation for <SCCM_install_dir>.
  - `<SCCM_install_dir>/bin/sccmProductInfo.sh`

Disabling Internet Explorer Enhanced Security Configuration

Internet Explorer Enhanced Security Configuration is an option that is provided in Windows Server 2003 operating systems and above. To use IBM SmartCloud Cost Management with Internet Explorer Version 7, you must disable Internet Explorer Enhanced Security Configuration.

About this task

When Internet Explorer Enhanced Security Configuration is enabled, it can create problems in viewing charts and some portlets. Follow these steps to disable Internet Explorer Enhanced Security Configuration:

Procedure

1. Close all instances of Internet Explorer.
2. Click **Start** > **Settings** > **Control Panel** and open **Add or Remove Programs**.
3. In the left panel of the Add or Remove Programs window, click **Add/Remove Windows Components**.
4. In the Windows Components Wizard dialog that is displayed, in the **Components** panel, select the **Internet Explorer Enhanced Security Configuration** entry and click **Details**.
5. In the Internet Explorer Enhanced Security Configuration dialog that is displayed, clear the check boxes for the listed user groups and click **OK**.
6. In the Windows Components Wizard dialog, click **Next** and once your settings have been applied, click **Finish**.

Results

Internet Explorer Enhanced Security Configuration is disabled.

Resolving the `FileNotFound Exception` error on UNIX and Linux systems

When a lot of files are open in IBM SmartCloud Cost Management you may encounter a `FileNotFound Exception` error message. This problem arises only for computers running UNIX or Linux operating systems.

About this task

This is a known issue with WebSphere Application Server environments, for more details see http://www-01.ibm.com/support/docview.wss?uid=swg21067352.

In relation to a particular IBM SmartCloud Cost Management instance, carry out the following steps to resolve the issue:

Procedure

1. Open the following file in a text editor:
   - `/etc/security/limits.conf`
2. Add the following lines to `limits.conf` and save the updated file:

   ```
   * soft nofile 32768
   * hard nofile 65536
   ```
3. Restart the computer.

Results

The `FileNotFound Exception` issue is now resolved.

Chapter 9. Reference

This reference information is organized to help you locate particular facts quickly.

Encryption information

IBM SmartCloud Cost Management uses various encryption algorithms for communication and storage of credentials.

For communication with other systems, https (SSL) is used, as defined in IETF RFC 5246.

For encryption of passwords for data sources in the `registry.tuam.xml` file, triple DES (3DES, 168 bit) is used.

FIPS Compliance

IBM SmartCloud Cost Management Version 2.1.0.5 supports the US Federal Information Processing Standard 140-2 (FIPS 140-2) when using cryptographic algorithms to encrypt and decrypt passwords. Password encryption and decryption is the only processing in SmartCloud Cost Management that is supported by FIPS.

For more information about FIPS, see http://csrc.nist.gov/publications/fips/

The following topics describe the areas in the SmartCloud Cost Management system in which passwords are encrypted and decrypted.

Enabling FIPS on the Application Server

You can configure the Application Server to use a Federal Information Processing Standard (FIPS) approved cryptographic provider.

Procedure

1. Modify the `java.security` file of the JRE used by the Administration Console. This file is located either in:

   ```
   <SCCM_install_dir>/jre/jre/lib/security/java.security
   ```

   OR

   ```
   /opt/ibm/java-x86_64-70/jre/lib/security/java.security
   ```

   Add a new **`security.provider`** line for **`com.ibm.crypto.provider.IBMJCEFIPS`** above the existing line for **`com.ibm.crypto.provider.IBMJCE`** and renumber all the existing provider lines in sequence. For example:

   ```
   #
   # List of providers and their preference orders (see above):
   #
   security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
   security.provider.2=com.ibm.crypto.provider.IBMJCEFIPS
   security.provider.3=com.ibm.crypto.provider.IBMJCE
   security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
   security.provider.5=com.ibm.security.cert.IBMCertPath
   security.provider.6=com.ibm.security.sasl.IBMSASL
   security.provider.7=com.ibm.xml.crypto.IBMXMLCryptoProvider
   ```

© Copyright IBM Corp. 2013, 2015

```
security.provider.8=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.10=sun.security.provider.Sun
```

Also, add 3 new lines to the file:

```
com.ibm.jsse2.JSSEFIPS=true
ssl.SocketFactory.provider=com.ibm.jsse2.SSLSocketFactoryImpl
ssl.ServerSocketFactory.provider=com.ibm.jsse2.SSLServerSocketFactoryImpl
```

and then restart the Application Server.

2. If you are using an SSL certificate that was not issued by a trusted CA, you must import the certificate into the Java keystore file. This can be done using the **keytool** command as follows:

```
keytool -keystore /opt/ibm/java-x86_64-70/jre/lib/security/cacerts -importcert -alias appserver -
```

OR

```
keytool -keystore SCCM_install_dir/jre/jre/lib/security/cacerts -importcert -alias appserver -fil
```

where **appserver.crt** is the SSL certificate that is used by the Application Server.

Enabling Tivoli Common Reporting for FIPS

Configure Cognos® to use the United States government Federal Information Processing Standard (FIPS) relating to encryption.

For information about how to enable Tivoli Common Reporting for FIPS, refer to the Enabling the reporting engine for FIPS topic in the Jazz for Service Management knowledge center.

Setting the SmartCloud Cost Management processing path

Use this task to verify that your processing path is correct.

Open the `config.properties` file which is located in `<SCCM_install_dir>/config` and edit the required properties. See related topic for more information.

REST API reference

Use these topics to learn more about the functionality and features of the representational state transfer (REST) application programming interface (API).

Note: 9080 is the default nonsecure port number for the administrative console and 9443 is the default secure port number. If your environment was configured with a port number other than the default, enter that number instead. If you are not sure of the port number, read the `server.xml` file to get the correct number.

Symbols and abbreviated terms

Certain symbols and abbreviated terms are used in the REST section. This topic explains those symbols and abbreviations.

ABNF Augmented Backus-Naur Form, as defined in RFC 5234.

HTTP Hyper Text Transfer Protocol. HTTP version 1.1 is defined in RFC 2616. Unless otherwise noted, the term HTTP is used in this document to mean both HTTP and HTTPS.

HTTPS
Hyper Text Transfer Protocol Secure, as defined in RFC 2818.

JSON JavaScript Object Notation, as defined in ECMA-262.

MIME Multipurpose Internet Mail Extensions, as defined in IANA MIME Media Types.

REST Representational State Transfer, as originally and informally described in Architectural Styles and the Design of Network-based Software Architectures.

URI Uniform Resource Identifier, as defined in RFC 3986.

XML eXtensible Markup Language, as defined by W3C.

REST API reference overview

This topic describes the representational state transfer (REST) application programming interface (API) provided by SmartCloud Cost Management.

Before you begin

The SmartCloud Cost Management application exposes a subset of its functionality using a REST API. By default, SmartCloud Cost Management exposes a REST API interface and there are no special configuration settings to enable or disable this interface. The SmartCloud Cost Management REST API is available on the same IP address or host name used to access the main application GUI. The application uses a self-signed certificate for its Secure Socket Layer (SSL) sessions. The same certificate is used for both the GUI and REST API sessions. You must configure your HTTPS client to either accept or ignore this certificate during the SSL handshake. You must use an HTTPS client that allows you to set the HTTP headers for each request. This is because there are multiple headers required for authentication, authorization, and content negotiation. For more details on how to work with SSL certificates in the liberty server see the *Securing the Liberty profile and its applications* topic in the knowledge center at http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wlp.nd.doc/ae/twlp_sec.html.

When generating HTTP requests to the SmartCloud Cost Management REST API, the following headers must be examined carefully:

Accept
The REST API generates both XML and JavaScript Object Notation (JSON) -encoded data in its responses. Include an `Accept: application/vnd.ibm.TUAM+xml` or `Accept: application/vnd.ibm.TUAM+json` header in your request to indicate the ability of your client to handle either XML or JSON responses.

Authentication
The REST API supports HTTP basic authentication. After successfully

authenticating, the server will return a cookie named **LtpaToken2** that is included with subsequent HTTP requests that are part of the same session. The **LtpaToken2** security token provides the capability to use a previously authenticated identity as part of starting the API. The same user IDs and passwords used to access the main SmartCloud Cost Management GUI are used to access the REST API.

Authorization

The authorization of a user to perform actions on the application is independent of the interface used to request the actions. The same authorization that applies for the SmartCloud Cost Management Administration Console applies to the REST API. Refer to the related concept about Configuring Keystone as a Central User Registry for more information.

Content-Type

All the content included in the HTTP request body sent to the application must be XML or JSON encoded. You must include either a `Content-Type: application/vnd.ibm.TUAM+xml` or `Content-Type: application/vnd.ibm.TUAM+json` header to indicate what encoded method is used for each API request that includes SmartCloud Cost Management payload data in XML or JSON formats.

Content Negotiation and Media Types

The SmartCloud Cost Management REST API supports multiple types of resource representations that are chosen through HTTP content negotiation. The supported types of resource representations in SmartCloud Cost Management include XML and JSON. The SmartCloud Cost Management REST API produces SmartCloud Cost Management XML and JSON format representations and as such new custom media types have been created to represent this. The client/server contract applies to content negotiation between these custom media types.

Some examples of the custom media types that SmartCloud Cost Management uses to support content negotiation between SmartCloud Cost Management XML and JSON format representations are as follows:

- application/vnd.ibm.TUAM+xml;version=1;format=client
- application/vnd.ibm.TUAM+json;version=1;format=client
- application/vnd.ibm.TUAM+xml;version=1;format=userGroup
- application/vnd.ibm.TUAM+json;version=1;format= userGroup

The SmartCloud Cost Management REST API uses the media type parameter `version=x` to represent version distinctions within a family of compatible and supported versions. The SmartCloud Cost Management REST API uses media type parameter `format=xyz` to represent format distinctions within a family of compatible formats.

The only top-level resources a client must know in the SmartCloud Cost Management REST API are the resource identifier (URIs) of the main domain entities that the client wants to interact with. For example, Clients, Users, AccountCodeStructures, and so on. SmartCloud Cost Management REST API operations allow retrieval of the resource identifiers of any REST resources representing Resource Types, with links returned along with previously returned resources.

Error messages

HTTP status codes are used to show errors. The definition of HTTP status codes defined in RFC2616 is the basis for each operation, and the operation

description may specify additional constraints on the use of HTTP status codes. Along with the status codes the SmartCloud Cost Management REST API provides additional error messages in the responses so that the client can establish what the problem relates to. These error messages have unique message IDs and message text explaining the error.

Operations

This section defines the operations of the protocol.

Overview on REST resources and HTTP methods

The operations of the SmartCloud Cost Management REST API protocol are defined as HTTP methods on certain REST resources, as listed in *Table 1*.

Table 177. Example of REST resources and HTTP methods

Target REST Resource	HTTP Methods
/rest/clients	GET
	POST
	PUT
	DELETE
/rest/users	GET
	POST
	PUT
	DELETE
/rest/usergroups	GET
	POST
	PUT
	DELETE
/rest/accountCodeStructures	GET
	POST
	PUT
	DELETE

HTTP Success and failure

The SmartCloud Cost Management REST API client is notified of success or failure by the HTTP status codes.

Common HTTP status codes that are used by the SmartCloud Cost Management Resources include:

```
200 (OK)
201 (Created)
400 (Bad Request)
404 (Not Found)
405 (Method Not Allowed)
406 (Not Acceptable)
409 (Conflict)
415 (Unsupported Media Type)
500 (Internal Server Error)
```

Links

Table 2 provides an overview of the links included in payload elements, and the resources that are targeted by these links.

Table 178. Example of Links included in payload elements

Payload element	Links included	Resource targeted by link
client	<clients><client> <uri>https://serverName:port /clients/12345</uri>	client
user	<users><user> <uri>https://serverName:port /users/JoeBlogs</uri>	user
accountCodeStructure	<accountCodeStructures> <accountCodeStructure> <uri>https://serverName:port /accountCodeStructures/Standard </uri>	accountCodeStructure
usergroup	<usergroups><usergroup> <uri>https://serverName:port /usergroups/TuamDev</uri>	usergroup
	<usergroups><usergroup><users> <user><uri>https://serverName:port /users/12345</uri>	user
	<usergroups><usergroup><clients> <client><uri>https://serverName: port/clients/12345</uri>	client
	<usergroups><usergroup> <accountCodeStructures> <accountCodeStructure> <uri>https://serverName:port /accountCodeStructures/Standard </uri>	accountCodeStructure

Common Request Header Behaviors

Accept

- Non Required Request Header for GET, PUT. If not specified by default the SmartCloud Cost Management server returns an XML payload with Media Type `application/vnd.ibm.TUAM+xml`
- Expected values -SmartCloud Cost Management Custom Media Type (`application/vnd.ibm.TUAM+xml` or `application/vnd.ibm.TUAM+json`). If an unexpected value is received, then the server returns an HTTP status code of `406` (Not Acceptable).

Optional Media Type Parameters include:

- **version** - API representation version from a family of compatible and supported versions.
- **format** - API representation format distinctions within a family of compatible formats

Content-Type

- Required Request Header for POST and PUT. If not specified by default for these methods the SmartCloud Cost Management server returns an HTTP status code of `415` (Unsupported Media Type).
- Expected values - SmartCloud Cost Management Custom Media Type (`application/vnd.ibm.TUAM+xml` or `application/vnd.ibm.TUAM+json`).

Optional Media Type Parameters include:

- **version** - API representation version from a family of compatible and supported versions.
- **format** - API representation format distinctions within a family of compatible formats

Protocol resource elements

This section describes the payload elements that are used in the payload of the messages of the operations.

Client resource element

A client resource element includes the properties that are displayed in *Table 1*.

Table 179. Properties of client resource element

Property name	Generic type	Requirement	Description
`accountCode`	String	Mandatory	Uniquely Identifies an entity for billing and reporting. This field is unique for the SmartCloud Cost Management database instance.
`accountName`	String	Optional	The name of the client as it is displayed in invoices and other reports.
`altAcctCode`	String	Optional	An alternative account code identification number (ID), that is used for reporting on data from another system in SmartCloud Cost Management, for example, a General Ledger system, where the standard account code is different.
`rateTable`	String	Optional	The rate table that is used for calculating the client charges.
`subscription`	subscription	Optional	The subscription that is derived from the `rateTable`. This is a read only property and is generated in GET requests only.

User resource element

A user resource element includes the properties that are displayed in *Table 2*.

Table 180. Properties of user resource element

Property name	Generic type	Requirement	Description
`userId`	String	Mandatory	A unique identifier for the user. This field is unique for the SmartCloud Cost Management database instance.
`userFullName`	String	Optional	A meaningful description of the user.
`emailToAddress`	String	Optional	User's email address.
`ldapUserId`	String	Optional	An LDAP user ID for the user if they have one. This id is used to log in to the SmartCloud Cost Management Administration Console and is also required for account code report security. If it is not defined in the request, it will be set to the `UserId`.

Usergroup resource element

A `usergroup` resource element includes the properties displayed in *Table 3*.

Table 181. Properties of usergroup resource element

Property name	Generic type	Requirement	Description
`groupId`	String	Mandatory	A unique identifier for the `usergroup`. This field is unique for the SmartCloud Cost Management database instance.
`groupFullName`	String	Mandatory	A description of the `usergroup`.
`users`	List <User>	Optional	All the users that are members of the `usergroup`.
`clients`	List <Client>	Optional	All the clients that are associated with the `usergroup`.
`accountStructures`	List < AccountStructure>	Optional	All Account Code Structures available to users in the group in SmartCloud Cost Management Reporting.

AccountCodeStructure resource element

An `accountCodeStructure` resource element includes the properties that are displayed in *Table 4*:

Table 182. Properties of accountCodeStructure resource element

Property name	Generic type	Requirement	Description
`accountCodeStructure`	String	Mandatory	Account Code Structure uniquely identifies an account code structure. This field is unique for the SmartCloud Cost Management database instance.
`startPosition`	Integer	Mandatory	Starting Offset into Account Code. Determines the account code level. If you want to begin from a level other than Level 1, enter the offset position for the level. The `startPosition` must be greater than one and less than 127.
`accountLevels`	List <AccountLevel>	Mandatory	All Account Levels associated with this property name`accountCodeStructure`. At least 1 level must be specified for an `accountCodeStructure`.

AccountLevel resource element

An `AccountLevel` resource element includes the properties that are displayed in *Table 5*.

Table 183. Properties of accountLevel resource element

Property name	Generic type	Requirement	Description
`accountLevelOrder`	Integer	Mandatory	The hierarchical level number that indicates where this level is displayed in the Account Code Structure.
`accountDescription`	String	Mandatory	Description of what the level represents.
`accountLength`	Integer	Optional	The number of characters the level uses in the account code. The full length of the account code structure is the sum of all the `accountLevel` lengths and cannot exceed the maximum length of 127.

Using Apache Wink to access SmartCloud Cost Management REST Resources

The Apache Wink client framework provides a Java API for implementing clients to use HTTP-based RESTful web services. The Apache Wink client framework also has a customizable handler mechanism to manipulate HTTP requests and responses. Apache Wink clients are built on JAX-RS principles and encompass REST concepts and standards.

Features of the Apache Wink client

By mapping REST-driven ideas to Java classes, they help to develop clients for Apache Wink-based services and any HTTP-driven RESTful web service. This makes them useful as a stand-alone REST-based Java client framework. The following are the main features of the Apache Wink Client:

- Serializes and de-serializes resources using configurable JAX-RS providers.
- Provides built-in support for various content types, including Atom, JSON, RSS, APP, CSV, and Multipart, along with corresponding object models in Java.
- Provides support for Secure Sockets Layer (SSL) and HTTP proxies.
- The client framework has a configurable handler chain for manipulating HTTP requests and responses.
- The client framework uses the `java.net.HttpUrlConnection` class for its core HTTP transport by default.
- Allows for easy customization of the core HTTP transport mechanism

The Apache Wink client framework is composed of the `RestClient` class, that acts as the central point of entry, holding the various configuration and provider registries. To start working with the client framework, it is necessary to create an instance of the `RestClient` object. Then, you create an instance of the Resource class from the new instance of the `RestClient` object using the URI of the SmartCloud Cost Management service you want to connect to. The Resource class is the Java equivalent of the RESTful web resource associated with the specific URI and is used to perform HTTP-based operations on it. All HTTP methods are filtered through a customizable handler chain that enables easy manipulation of HTTP requests and responses.

GET requests

The code example demonstrates the use of an HTTP GET on a SmartCloud Cost Management Client Resource request using the Apache Wink client.

```
/*
 * Create a ClientConfig Object and attach an authenticated
 * BasicAuthSecurityHandler to it.
 */
ClientConfig config = new ClientConfig();
BasicAuthSecurityHandler basicAuthHandler = new BasicAuthSecurityHandler();
basicAuthHandler.setUserName("userName");
basicAuthHandler.setPassword("password");
config.handlers(basicAuthHandler);

//Create the rest client instance with a
RestClient libraryClient = new RestClient(config);

String resoruceUri = "http://localhost:8881/tuamConsole/rest/clients";

/*
 * Create the resource instance to interact with, indicating where it's located
```

```
 * with the URL
 */
Resource clientResource = libraryClient.resource(resoruceUri);

//Add a Http Accept header with expected format
clientResource.header("Accept", "application/xml");

// perform a GET on the resource. The resource will be returned as an xml String
String xmlClient = clientResource.get().getEntity(String.class);
```

The `RestClient` object is the entry point for the Apache Wink client framework. To build a RESTful web service client, you must create an instance of the `RestClient` object. When the `RestClient` object is created a new Resource object is created with the URI of the service you want to start by calling the `RestClient#resource()` method. To issue an HTTP GET request, you must start the `Resource#get()` method. This method returns the HTTP response. Then, by starting the appropriate provider, the client can de-serialize the response.

POST request

The following code example demonstrates the use of an HTTP POST request using the Apache Wink client.

```
/*
 * Create a ClientConfig Object and attach an authenticated
 * BasicAuthSecurityHandler to it.
 */
ClientConfig config = new ClientConfig();
BasicAuthSecurityHandler basicAuthHandler = new BasicAuthSecurityHandler();
basicAuthHandler.setUserName("userName");
basicAuthHandler.setPassword("password");
config.handlers(basicAuthHandler);

//Create the rest client instance with a
RestClient libraryClient = new RestClient(config);

String resoruceUri = "http://localhost:8881/tuamConsole/rest/clients";

/*
 * Create the resource instance to interact with, indicating where it's located
 * with the URL
 */
Resource clientResource = libraryClient.resource(resoruceUri);

//Create an XML message String
String message =
 "<client>"+
 "    <accountCode>accountCode0</accountCode>"+
 "    <accountName>pAccountName0</accountName>"+
 "    <altAcctCode>altAccountCode0</altAcctCode>"+
 "    <rateTable>STANDARD</rateTable>"+
 "</client>";

// Set the Http contentType header of the requet and perform a POST on the
resource.
ClientResponse clientResponse =  clientResource.contentType("application/xml").
post(message);
```

When you issue the POST request, it is similar to issuing a GET request. Just like the GET request example, you must create an instance of a Resource through the `RestClient`. The only difference is that the POST string is passed in as a method parameter to the `resource.post` method after specifying the request media type.

Clients Rest API

You can use the representational state transfer (REST) application programming interface (API) to manage `clients`.

GET Resource clients

Use the GET Resource clients to get all the SmartCloud Cost Management clients defined

Table 184. GET Resource clients

REST resource elements	**Details**
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/clients`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Request payload	None
Request Content-Type	None
Response headers	Content-Type

Table 184. GET Resource clients (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:clients xmlns:ns="http://www.ibm.com /xmlns/prod/TUAM/1.0">` ` <client accountCode="ATM" >` `  <uri>..... /clients/ATM</ uri >` `  <accountName>ATM Transactions </accountName>` `  <altAcctCode>2000-3000-4000</altAcctCode>` `  <rateTable>STANDARD</rateTable>` `  <subscription id="STANDARD">` `  <uri>../subscriptions/STANDARD</uri>` `  </subscription>` ` </client>` `  <client accountCode= "CCX" >` `  <uri>..... /clients/ CCX</uri>` `  <accountName>Credit Card</accountName>` `  <altAcctCode>2000-3000-5000</altAcctCode>` `  <rateTable>STANDARD</rateTable>` `  <subscription id="STANDARD">` `  <uri>../subscriptions/STANDARD</uri>` `  </subscription>` ` </client>` `<ns:clients>` **Response JSON** `[` ` {"uri": "..... /clients/ATM"` `  "accountCode": "ATM",` `  "accountName": "ATM Transactions",` `  "altAcctCode": "2000-3000-4000",` `  "rateTable": "STANDARD"` `  "subscription": {` `  "id": "STANDARD",` `  "uri": "../subscriptions/STANDARD"` `  }` ` },` `  {` `  "uri": "..... /clients/CCX"` `  "accountCode": "CCX",` `  "accountName": "Credit Card",` `  "altAcctCode": "2000-3000-5000",` `  "rateTable": "STANDARD"` `  "subscription": {` `  "id": "STANDARD",` `  "uri": "../subscriptions/STANDARD"` `  }` ` }` `]`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Generic operation	None
Normal HTTP response codes	**200**: Returns the list of all clients defined in SmartCloud Cost Management.

Table 184. GET Resource clients (continued)

REST resource elements	Details
Error HTTP response codes	**403**: This code is returned if the requester does not have sufficient permission to view the list of users. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request

GET{id} Resource clients

Use the GET{id} Resource clients to get a single SmartCloud Cost Management client defined by `accountCode`.

Table 185. GET Resource clients

REST resource elements	Details
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/clients/{id}`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Request payload	None
Request Content-Type	None
Response headers	Content-Type
Response payload	**Response XML** `<ns:clients xmlns:ns="http://www.ibm.com/xmlns/prod/TUAM/1.0 "accountCode="ATM" >` `<uri>..... /clients/ATM</uri>` `<accountName>ATM Transactions</accountName>` `<altAcctCode>2000-3000-4000</altAcctCode>` `<rateTable>STANDARD</rateTable>` `<subscription id="STANDARD">` `<uri>../subscriptions/STANDARD</uri>` `</subscription>` `</ns:client>` **Response JSON** `{` `"uri": "..... /clients/ATM"` `"accountCode": "ATM",` `"accountName": "ATM Transactions",` `"altAcctCode": "2000-3000-4000",` `"rateTable": "STANDARD"` `"subscription": {` `"id": "STANDARD",` `"uri": "../subscriptions/STANDARD"` `}` `}`

Table 185. GET Resource clients (continued)

REST resource elements	Details
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Generic operation	None
Normal HTTP response codes	**200**: Returns information about a client defined in SmartCloud Cost Management.
Error HTTP response codes	**400**: This code is returned if the `accountCode` does not exist in the request. **403**: This code is returned if the requester does not have permission to view the client. **404**: This code is returned if the requested client is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

POST Resource clients

Use the POST Resource clients to create a SmartCloud Cost Management client.

Table 186. Post Resource clients

REST resource elements	Details
HTTP method	POST
Resource URI	`https://serverName:port/tuamConsole/rest/clients`
URI query parameters	None
Request headers	Content-Type
Request payload	**Request XML** `<ns:clients xmlns:ns="http://www.ibm.com/xmlns/prod/TUAM/1.0 "  accountCode="ATM" >` `  <accountName>ATM Transactions</accountName>` `  <altAcctCode>2000-3000-4000</altAcctCode>` `  <rateTable>STANDARD</rateTable>` `</ns:client>` **Request JSON** `{` `  "uri": "..... /clients/ATM"` `  "accountCode": "ATM",` `  "accountName": "ATM Transactions",` `  "altAcctCode": "2000-3000-4000",` `  "rateTable": "STANDARD"` `}`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Response headers	Location: The URL of the new client is included in the Location header of the response.
Response payload	None

Table 186. Post Resource clients (continued)

REST resource elements	Details
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**201**: The client is defined as requested. The URL of the new client is included in the Location header of the response.
Error HTTP response codes	**400**: This code is returned in the following circumstances: • if the `accountCode` is not passed in the request. • if the `rateTable` passed in the request doesn't exist **403**: This code is returned if the requester does not have sufficient permission to define a new client. **409**: This code is returned if the client passed in the request already exists. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

PUT Resource clients

Use the PUT Resource clients to replace the representation of an existing SmartCloud Cost Management client.

Table 187. PUT Resource clients

REST resource elements	Details
HTTP method	PUT
Resource URI	`https://serverName:port/tuamConsole/rest/clients`
URI query parameters	None
Request headers	Content-Type, Accept
Request payload	**Request XML** `<ns:clients xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0 " accountCode="ATM" > <accountName>ATM Transactions</accountName> <altAcctCode>2000-3000-4000</altAcctCode> <rateTable>STANDARD</rateTable> </ns:clients>` **Request JSON** `{ "uri": "..... /clients/ATM" "accountCode": "ATM", "accountName": "ATM Transactions", "altAcctCode": "2000-3000-4000", "rateTable": "STANDARD" }`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Response headers	Content-Type

Table 187. PUT Resource clients (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:clients xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0 " accountCode="ATM" > <uri>..... /clients/ATM</uri> <accountName>ATM Transactions</accountName> <altAcctCode>2000-3000-4000</altAcctCode> <rateTable>STANDARD</rateTable> </ns:client>` **Response JSON** `{ "uri": "..... /clients/ATM" "accountCode": "ATM", "accountName": "ATM Transactions", "altAcctCode": "2000-3000-4000", "rateTable": "STANDARD" }`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=client` `orapplication/vnd.ibm.TUAM+json;version=1;format=client`
Generic operation	None
Normal HTTP response codes	**200**: The client has been successfully modified. The new Client is included in the response.
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `accountCode` is not passed in the request. • if the `rateTable` passed in the request doesn't exist. **403**: This code is returned if the requester does not have permission to update the specified client. **404**: This code is returned if the requested client is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

DELETE Resource clients

Use the DELETE Resource clients to delete the representation of an existing SmartCloud Cost Management client for a given `accountCode`.

Table 188. DELETE Resource clients

REST resource elements	Details
HTTP method	DELETE
Resource URI	`https://serverName:port/tuamConsole/rest/clients/{id}`
URI query parameters	None
Request headers	None
Request payload	None

Table 188. DELETE Resource clients (continued)

REST resource elements	Details
Request Content-Type	None
Response headers	None
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**204**: The client has been deleted as requested.
Error HTTP response codes	**400**: This code is returned if the `accountCode` is not passed in the request. **403**: This code is returned if the requester does not have permission to delete the client. **404**: This code is returned if the requested client is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

Users REST API

You can use the representational state transfer (REST) application programming interface (API) to manage `users`.

GET Resource users

Use the GET Resource users to get all the SmartCloud Cost Management users defined

Table 189. GET Resource users

REST resource elements	Details
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/users`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format=user` `orapplication/vnd.ibm.TUAM+json;version=1;format=user`
Request payload	None
Request Content-Type	None
Response headers	Content-Type

Table 189. GET Resource users (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:users xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" >` `  <user userID=" UserID1">` `  <uri>..... /users/UserID1</uri>` `<userFullName>UserFullName1</userFullName>` `<emailToAddress>test@ibm.com</emailToAddress>` `  </user>` `  <user userID="UserID2">` `  <uri>..... /users/UserID2</uri>` `<userFullName>UserFullName2</userFullName>` `<emailToAddress>test1@ibm.com</emailToAddress>` `  </user>` `</ns:users>` **Response JSON** `[` `{` `" uri ":" ..... /users/UserID1",` `"userFullName":"UserFullName1",` `"userID":"UserID1"` `},` `{` `" uri ":" ..... /users/UserID1",` `"userFullName":"UserFullName2",` `"userID":"UserID2"` `}` `]`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=user` `orapplication/vnd.ibm.TUAM+json;version=1;format= user`
Generic operation	None
Normal HTTP response codes	**200**: Returns the list of all users defined in SmartCloud Cost Management.
Error HTTP response codes	**403**: This code is returned if the requester does not have sufficient permission to view the list of users. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request

GET{id} Resource users

Use the GET{id} Resource users to get a single SmartCloud Cost Management user defined by userId {id}

Table 190. GET{id} Resource users

REST resource elements	Details
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/users/{id}`
URI query parameters	None

Table 190. GET{id} Resource users (continued)

REST resource elements	Details
Request headers	Accept application/vnd.ibm.TUAM+xml;version=1;format=user orapplication/vnd.ibm.TUAM+json;version=1;format=user
Request payload	None
Request Content-Type	None
Response headers	Content-Type
Response payload	**Response XML** <ns:user xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userID=" UserID1"> <uri>..... /users/UserID1</uri> <userFullName>UserFullName1</userFullName> <emailToAddress>test@ibm.com</emailToAddress> </ns:user> **Response JSON** { " uri ":" ..... /users/UserID1", "userFullName":"UserFullName2", "userID":"UserID2" }
Response Content-Type	application/vnd.ibm.TUAM+xml;version=1;format=user orapplication/vnd.ibm.TUAM+json;version=1;format= user
Generic operation	None
Normal HTTP response codes	**200**: Returns information about a user defined in SmartCloud Cost Management.
Error HTTP response codes	**400**: This code is returned if the userId does not exist in the request. **403**: This code is returned if the requester does not have permission to view the user. **404**: This code is returned if the requested user is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

POST Resource users

Use the POST Resource users to create a SmartCloud Cost Management user.

Table 191. POST Resource users

REST resource elements	Details
HTTP method	POST
Resource URI	https://serverName:port/tuamConsole/rest/users
URI query parameters	None
Request headers	Content-Type

Table 191. POST Resource users (continued)

REST resource elements	Details
Request payload	**Request XML** `<ns:user xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userID=" UserID1"> <userFullName>UserFullName1</userFullName> <emailToAddress>test@ibm.com</emailToAddress> </ns:user>` **Request JSON** `"userFullName":"UserFullName2", "userID":"UserID2" }`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=user` `orapplication/vnd.ibm.TUAM+json;version=1;format=user`
Response headers	Location: The URL of the new user is included in the Location header of the response.
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**201**: The user has been defined as requested. The URL of the new user is included in the Location header of the response.
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `userId` does not exist in the request. **403**: This code is returned if the requester does not have sufficient permission to define a new user. **409**: This code is returned if the user entered in the request already exists. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

PUT Resource users

Use the PUT Resource users to replace the representation of an existing SmartCloud Cost Management user.

Table 192. PUT Resource users

REST resource elements	Details
HTTP method	PUT
Resource URI	`https://serverName:port/tuamConsole/rest/users`
URI query parameters	None
Request headers	Content-Type, Accept

Table 192. PUT Resource users (continued)

REST resource elements	Details
Request payload	**Request XML** `<ns:user xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userID=" UserID1"> <userFullName>UserFullName1</userFullName> <emailToAddress>test@ibm.com</emailToAddress> </ns:user>` **Request JSON** `{ "userFullName":"UserFullName2", "userID":"UserID2" }`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=user` `orapplication/vnd.ibm.TUAM+json;version=1;format= user`
Response headers	Content-Type
Response payload	**Response XML** `<ns:user xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userID=" UserID1"> <userFullName>UserFullName1</userFullName> <emailToAddress>test@ibm.com</emailToAddress> </ns:user>` **Response JSON** `{ "userFullName":"UserFullName2", "userID":"UserID2" }`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=user` `orapplication/vnd.ibm.TUAM+json;version=1;format= user`
Generic operation	None
Normal HTTP response codes	**200**: The user has been successfully modified. The new User is included in the response.
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `userId` does not exist in the request. **403**: This code is returned if the requester does not have permission to update the specified user. **404**: This code is returned if the requested user is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

DELETE Resource users

Use the DELETE Resource users to delete the representation of an existing SmartCloud Cost Management user for a given `userId`.

Table 193. DELETE Resource users

REST resource elements	**Details**
HTTP method	DELETE
Resource URI	`https://serverName:port/tuamConsole/rest/users/{id}`
URI query parameters	None
Request headers	None
Request payload	None
Request Content-Type	None
Response headers	None
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**204**: The user has been deleted as requested.
Error HTTP response codes	**400**: This code is returned if the `userId` does not exist in the request. **403**: This code is returned if the requester does not have permission to delete the user. **404**: This code is returned if the requested user is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

Usergroups REST API

You can use the representational state transfer (REST) application programming interface (API) to manage `usergroups`.

GET Resource usergroups

Use the GET Resource `usergroups` to get all the SmartCloud Cost Management `usergroups` defined.

Table 194. GET Resource usergroups

REST resource elements	**Details**
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/ usergroups`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format= usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format= usergroup`
Request payload	None
Request Content-Type	None

Table 194. GET Resource usergroups (continued)

REST resource elements	Details
Response headers	Content-Type
Response payload	**Response XML** <ns:userGroups xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0"> <userGroups> <userGroup groupID="groupID"> <uri>..... userGroups/groupID</uri> <groupFullName>groupFullName</groupFullName> <users> <user userID="userID" > <uri>..... /users/userID</uri> </user> <user userID="userID1" > <uri>..... /users/userID1</uri> </user> </users> <clients> <client accountCode="accountCode1"> <uri>..... /clients/accountCode1</uri> </client> <client accountCode="accountCode2"> <uri>..... /clients/accountCode2</uri> </client> </clients> <accountStructures> <accountStructure accountStructureName ="name"> <uri>..... /accountStructures/name</uri> <groupDefault>true</groupDefault> </accountStructure> <accountStructure accountStructureName ="name1"> <uri>...../accountStructures/name1</uri> <groupDefault>false</groupDefault> </accountStructure> </accountStructures> </userGroup> </ns:userGroups> **Response JSON** [{ "users": [{" uri ":" ..... /users/userID", "userId": "userID"}, {" uri ":" ..... /users/userID1", "userId": "userID1"}], "clients": [{" uri ":" ..... /clients/accountCode1", "accountCode": "accountCode1"}, {" uri ":" ..... /clients/accountCode2", "accountCode": "accountCode2"}], "accountStructures": [{" uri ":" ..... /accountStructures/name", "accountStructureName": "name", "groupDefault": true}, {" uri ":" ..... /accountStructures/name1", "accountStructureName": "name1", "groupDefault": true}], "groupFullName": "groupFullName1", "groupID": "groupID1" " uri ":" ..... /usergroups/groupID1", }]

Table 194. GET Resource usergroups (continued)

REST resource elements	Details
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=usergroups` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroups`
Generic operation	None
Normal HTTP response codes	**200**: Returns the list of all `usergroups` defined in SmartCloud Cost Management.
Error HTTP response codes	**403**: This code is returned if the requester does not have sufficient permission to view the list of `usergroups`. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

Get{id} Resource usergroups

Use the `GET{id}` Resource `usergroups` to get a single SmartCloud Cost Management user defined by `groupId {id}`.

Table 195. GET{id} Resource usergroups

REST resource elements	Details
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/usergroups/{id}`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format=usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroup`
Request payload	None
Request Content-Type	None
Response headers	Content-Type

Table 195. GET{id} Resource usergroups (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:userGroup xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userGroup groupID="groupID"> <uri>..... userGroups/groupID</uri> <groupFullName>groupFullName </groupFullName> <users> <user userID="userID" > <uri>..... /users/userID</uri> </user> <user userID="userID1" > <uri>..... /users/userID1</uri> </user> </users> <clients> <client accountCode="accountCode1"> <uri>..... /clients/accountCode1</uri> </client> <client accountCode="accountCode2"> <uri>..... /clients/accountCode2</uri> </client> </clients> <accountStructures> <accountStructure accountStructureName ="name"> <uri>..... /accountStructures/name</uri> <groupDefault>true</groupDefault> </accountStructure> <accountStructure accountStructureName ="name1"> <uri>...../accountStructures/name1</uri> <groupDefault>false</groupDefault> </accountStructure> </accountStructures> </ ns:userGroup>`  **Response JSON** `{ "users": [{" uri ":" ..... /users/userID", "userId": "userID"}, {" uri ":" ..... /users/userID1", "userId": "userID1"}], "clients": [{" uri ":" ..... /clients/accountCode1", "accountCode": "accountCode1"}, {" uri ":" ..... /clients/accountCode2", "accountCode": "accountCode2"}], "accountStructures": [{" uri ":" ..... /accountStructures/name", "accountStructureName": "name", "groupDefault": true}, {" uri ":" ..... /accountStructures/name1", "accountStructureName": "name1", "groupDefault": true}], "groupFullName": "groupFullName1", "groupID": "groupID1" " uri ":" ..... /usergroups/groupID1", }`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroup`
Generic operation	None

Table 195. GET{id} Resource usergroups (continued)

REST resource elements	Details
Normal HTTP response codes	**200**: Returns information about a `usergroup` defined in SmartCloud Cost Management.
Error HTTP response codes	**400**: This code is returned if the `groupId` does not exist in the request. **403**: This code is returned if the requester does not have permission to view the user. **404**: This code is returned if the requested `usergroup` is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

POST Resource usergroups

Use the POST Resource `usergroups` to create a SmartCloud Cost Management `usergroup`.

Table 196. POST Resource usergroups

REST resource elements	Details
HTTP method	POST
Resource URI	`https://serverName:port/tuamConsole/rest/usergroups`
URI query parameters	None
Request headers	Content-Type

Table 196. POST Resource usergroups (continued)

REST resource elements	Details
Request payload	**Request XML** `<ns:userGroup xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" groupID="groupID">` `<groupFullName>groupFullName</groupFullName>` `<users>` `<user userID="userID" >` `</user>` `<user userID="userID1" >` `</user>` `</users>` `<clients>` `<client accountCode="accountCode1">` `</client>` `<client accountCode="accountCode2">` `</client>` `</clients>` `<accountStructures>` `<accountStructure accountStructureName ="name">` `<groupDefault>true</groupDefault>` `</accountStructure>` `<accountStructure accountStructureName ="name1">` `<groupDefault>false</groupDefault>` `</accountStructure>` `</accountStructures>` `</ns:userGroup>` **Request JSON** `{` `"users":` `[` `{"userId": "userID"},` `{"userId": "userID1"}` `],` `"clients":` `[` `{"accountCode": "accountCode1"},` `{"accountCode": "accountCode2"}` `],` `"accountStructures":` `[` `{"accountStructureName": "name",` `"groupDefault": true},` `{"accountStructureName": "name1",` `"groupDefault": true}` `],` `"groupFullName": "groupFullName1",` `"groupID": "groupID1"` `}`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroup`
Response headers	Location: The URL of the new `usergroup` is included in the Location header of the response.
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**201**: The `usergroup` is defined as requested. The URL of the new `usergroup` is included in the Location header of the response.

Table 196. POST Resource usergroups (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `groupId` does not exist in the request. • if the `groupFullName` does not exist in the request. • if the requested `AccountCodeStructures` related to the Group, does not specify a default `AccountCodeStructure`. **403**: This code is returned if the requester does not have sufficient permission to define a new user. **404**: This code is returned in the following cases: • if the requested User related to the Group, is not defined. • if the requested Client related to the Group, is not defined. • if the requested `AccountCodeStructure` related to the Group is not defined. **409**: This code is returned if the `usergroup` passed in the request already exists. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error when processing the request.

PUT Resource usergroups

Use the PUT Resource `usergroups` to replace the representation of an existing SmartCloud Cost Management `usergroup`.

Table 197. PUT Resource usergroups

REST resource elements	Details
HTTP method	PUT
Resource URI	`https://serverName:port/tuamConsole/rest/usergroups`
URI query parameters	None
Request headers	Content-Type, Accept

Table 197. PUT Resource usergroups (continued)

REST resource elements	Details
Request payload	**Request XML** `<ns:userGroup xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userGroup groupID="groupID"> <groupFullName>groupFullName</groupFullName> <users> <user userID="userID" > </user> <user userID="userID1" > </user> </users> <clients> <client accountCode="accountCode1"> </client> <client accountCode="accountCode2"> </client> </clients> <accountStructures> <accountStructure accountStructureName ="name"> <groupDefault>true</groupDefault> </accountStructure> <accountStructure accountStructureName ="name1"> <groupDefault>false</groupDefault> </accountStructure> </accountStructures> </ns:userGroup>`  **Request JSON** `{ "users": [{"userId": "userID"}, {"userId": "userID1"}], "clients": [{"accountCode": "accountCode1"}, {"accountCode": "accountCode2"}], "accountStructures": [{"accountStructureName": "name", "groupDefault": true}, {"accountStructureName": "name1", "groupDefault": true}], "groupFullName": "groupFullName1", "groupID": "groupID1" }`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroup`
Response headers	Content-Type

Table 197. PUT Resource usergroups (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:userGroup xmlns:ns="http://www.ibm.com/xmlns /prod/TUAM/1.0" userGroup groupID="groupID"> <uri>..... userGroups/groupID</uri> <groupFullName>groupFullName</groupFullName> <users> <user userID="userID" > <uri>..... /users/userID</uri> </user> <user userID="userID1" > <uri>..... /users/userID1</uri> </user> </users> <clients> <client accountCode="accountCode1"> <uri>..... /clients/accountCode1</uri> </client> <client accountCode="accountCode2"> <uri>..... /clients/accountCode2</uri> </client> </clients> <accountStructures> <accountStructure accountStructureName ="name"> <uri>..... /accountStructures/name</uri> <groupDefault>true</groupDefault> </accountStructure> <accountStructure accountStructureName ="name1"> <uri>...../accountStructures/name1</uri> <groupDefault>false</groupDefault> </accountStructure> </accountStructures> </ns:userGroup>`  **Response JSON** `{ "users": [{" uri ":" ..... /users/userID", "userId": "userID"}, {" uri ":" ..... /users/userID1", "userId": "userID1"}], "clients": [{" uri ":" ..... /clients/accountCode1", "accountCode": "accountCode1"}, {" uri ":" ..... /clients/accountCode2", "accountCode": "accountCode2"}], "accountStructures": [{" uri ":" ..... /accountStructures/name", "accountStructureName": "name", "groupDefault": true}, {" uri ":" ..... /accountStructures/name1", "accountStructureName": "name1", "groupDefault": true}], "groupFullName": "groupFullName1", "groupID": "groupID1" " uri ":" ..... /usergroups/groupID1", }`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=usergroup` `orapplication/vnd.ibm.TUAM+json;version=1;format=usergroup`
Generic operation	None

Table 197. PUT Resource usergroups (continued)

REST resource elements	Details
Normal HTTP response codes	**200**: The `usergroup` has been successfully modified. The new `usergroup` is included in the response.
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `groupId` does not exist in the request. • if the requested `AccountCodeStructures`, related to the Group does not specify a default `AccountCodeStructure` **403**: This code is returned if the requester does not have permission to update the specified `usergroup`. **404**: This code is returned in the following cases: • if the requested `usergroup` is not defined. • if the requested User related to the Group is not defined. • if the requested Client related to the Group is not defined. • if the requested `AccountCodeStructure` related to the Group is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

DELETE Resource usergroups

Use the DELETE Resource `usergroups` to delete the representation of an existing SmartCloud Cost Management user group for a given `groupId`.

Table 198. DELETE Resource usergroups

REST resource elements	Details
HTTP method	DELETE
Resource URI	`https://serverName:port/tuamConsole/rest/usergroups/{id}`
URI query parameters	None
Request headers	None
Request payload	None
Request Content-Type	None
Response headers	None
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**204**: The `usergroup` has been deleted as requested.

Table 198. DELETE Resource usergroups (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned if the `groupId` is not passed in the request. **403**: This code is returned if the requester does not have permission to delete the `usergroup`. **404**: This code is returned if the requested `usergroup` is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

AccountCodeStructrues REST API

You can use the representational state transfer (REST) application programming interface (API) to manage `accountCodeStructrues`.

GET Resource accountCodeStructrues

Use the GET `Resource accountCodeStructrues` to get all the SmartCloud Cost Management `accountCodeStructrues` defined.

Table 199. GET Resource accountCodeStructrues

REST resource elements	Details
HTTP method	GET
Resource URI	`https://serverName:port/tuamConsole/rest/accountCodeStructrues`
URI query parameters	None
Request headers	Accept `application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue` `orapplication/` `vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue`
Request payload	None
Request Content-Type	None
Response headers	Content-Type

Table 199. GET Resource accountCodeStructrues (continued)

REST resource elements	Details
Response payload	**Response XML** <ns:accountCodeStructures xmlns:ns=http://www.ibm.com /xmlns/prod/TUAM/1.0> <accountCodeStructure accountStructureName="STANDARD"> <startPosition>1</startPosition> <accountLevels> <accountLevel> <accountLevelOrder>1</accountLevelOrder> <accountLength>16</accountLength> <accountDescription>accountDescription </accountDescription> </accountLevel> <accountLevel> <accountLevelOrder>accountLevelOrder </accountLevelOrder> <accountLength>accountLength</accountLength> <accountDescription>accountDescription </accountDescription> </accountLevel> </accountLevels> <uri>....../accountCodeStructures/STANDARD</uri> </accountCodeStructure> <accountCodeStructure accountStructureName=" accountStructureName1"> <startPosition>16</startPosition> <accountLevels> <accountLevel> <accountLevelOrder>2</accountLevelOrder> <accountLength>32</accountLength> <accountDescription>accountDescription1 </accountDescription> </accountLevel> </accountLevels> <uri>...../accountCodeStructures /accountStructureName1</uri> </accountCodeStructure> </ns:accountCodeStructures>

Table 199. GET Resource accountCodeStructrues (continued)

REST resource elements	Details
Response payload (continued)	**Response JSON**[{ "accountLevels": [{ "accountDescription": "level1", "accountLength": 16, "accountLevelOrder": 1 }, { "accountDescription": "level2", "accountLength": 16, "accountLevelOrder": 2 }], "accountStructureName": "STANDARD", "startPosition": 0 "uri": "..../accountCodeStructures/STANDARD" }, { "accountLevels": [{ "accountDescription": "level1", "accountLength": 16, "accountLevelOrder": 1 }, { "accountDescription": "level2", "accountLength": 16, "accountLevelOrder": 2 }], "accountStructureName": "CLOUDSTANDARD", "startPosition": 0 "uri": "..../accountCodeStructures/STANDARD" }]
Response Content-Type	application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue orapplication/ vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue
Generic operation	None
Normal HTTP response codes	**200**: Returns the list of all users defined in SmartCloud Cost Management.
Error HTTP response codes	**403**: This code is returned if the requester does not have sufficient permission to view the list of users. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

GET{id} Resource accountCodeStructrues

Use the GET{id} Resource `accountCodeStructrues` to get a single SmartCloud Cost Management `accountCodeStructrue` defined by `accountStructureName` {id}.

Table 200. GET{id} Resource accountCodeStructrues

REST resource elements	Details
HTTP method	GET
Resource URI	https://serverName:port/tuamConsole/rest/accountCodeStructrues/{id}
URI query parameters	None

Table 200. GET{id} Resource accountCodeStructrues (continued)

REST resource elements	Details
Request headers	Accept application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue orapplication/ vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue
Request payload	None
Request Content-Type	None
Response headers	Content-Type
Response payload	**Response XML** <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:accountCodeStructure xmlns:ns2="http://www.ibm.com/xmlns /prod/tuam/1.0"accountStructureName="Standard"> <startPosition>1</startPosition> <accountLevels> <accountLevel> <accountLevelOrder>1</accountLevelOrder> <accountDescription>Application</accountDescription> <accountLevelLength>4</accountLevelLength> </accountLevel> <accountLevel> <accountLevelOrder>2</accountLevelOrder> <accountDescription>Resource group</accountDescription> <accountLevelLength>16</accountLevelLength> </accountLevel> <accountLevel> <accountLevelOrder>3</accountLevelOrder> <accountDescription>Platform</accountDescription> <accountLevelLength>16</accountLevelLength> </accountLevel> <accountLevel> <accountLevelOrder>4</accountLevelOrder> <accountDescription>Server</accountDescription> <accountLevelLength>20</accountLevelLength> </accountLevel> </accountLevels> <uri>https://localhost:16311/tuamConsole/rest/accountCodeStructures /STANDARD</uri></ns2:accountCodeStructure> **Response JSON** { "accountLevels": [{ "accountDescription": "level1", "accountLength": 16, "accountLevelOrder": 1 }, { "accountDescription": "level2", "accountLength": 16, "accountLevelOrder": 2 }], "accountStructureName": "STANDARD", "startPosition": 0 "uri": "..../accountCodeStructures/STANDARD" }
Response Content-Type	application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue orapplication/ vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue
Generic operation	None
Normal HTTP response codes	**200**: Returns information about an accountCodeStructrue defined in SmartCloud Cost Management.

Table 200. GET{id} Resource accountCodeStructrues (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned if the `accountCodeStructrue` does not exist in the request. **403**: This code is returned if the requester does not have permission to view the client. **404**: This code is returned if the requested client is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

POST Resource accountCodeStructrues

Use the POST Resource `accountCodeStructrues` to create a SmartCloud Cost Management `accountCodeStructrue`.

Table 201. POST Resource accountCodeStructrues

REST resource elements	Details
HTTP method	POST
Resource URI	`https://serverName:port/tuamConsole/rest/accountCodeStructrues`
URI query parameters	None
Request headers	Content-Type

Table 201. POST Resource accountCodeStructrues (continued)

REST resource elements	Details
Request payload	**Response XML** <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <ns2:accountCodeStructure xmlns:ns2="http://www.ibm.com/xmlns/prod/TUAM/1.0" accountStructureName="Standard"> <startPosition>1</startPosition> <accountLevels> <accountLevel> <accountLevelOrder>1</accountLevelOrder> <accountDescription>Application</accountDescription> <accountLevelLength>4</accountLevelLength> </accountLevel> <accountLevel><accountLevelOrder>2</accountLevelOrder> <accountDescription>Resource group</accountDescription> <accountLevelLength>16</accountLevelLength></accountLevel> <accountLevel><accountLevelOrder>3</accountLevelOrder> <accountDescription>Platform</accountDescription> <accountLevelLength>16</accountLevelLength> </accountLevel> <accountLevel><accountLevelOrder>4</accountLevelOrder> <accountDescription>Server</accountDescription> <accountLevelLength>20</accountLevelLength> </accountLevel> </accountLevels> <uri>https://localhost:16311/TUAMConsole/rest/accountCodeStructures/STANDARD</uri> </ns2:accountCodeStructure> **Response JSON** { "accountLevels": [{ "accountDescription": "level1", "accountLength": 16, "accountLevelOrder": 1 }, { "accountDescription": "level2", "accountLength": 16, "accountLevelOrder": 2 }], "accountStructureName": "STANDARD", "startPosition": 0 }
Request Content-Type	application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue orapplication/ vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue
Response headers	Content-Type Location: The URL of the new user is included in the Location header of the response.
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**201**: The accountCodeStructure has been defined as requested. The URL of the new accountCodeStructure is included in the Location header of the response.

Table 201. POST Resource accountCodeStructrues (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned in the following cases • if the `accountStructureName` does not exist in the request. • if the start position of the `accountCodeStructure` is less than one or greater that the maximum length. • if there is no level specified. • if any account level does not have a description. • if full length of the account code structure exceeds the max length. **403**: This code is returned if the requester does not have sufficient permission to define a new user. **409**: This code is returned if the `accountCodeStructure` passed in the request already exists **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

PUT Resource accountCodeStructrue

Use the PUT Resource `accountCodeStructrue` to replace the representation of an existing SmartCloud Cost Management `accountCodeStructrue`.

Table 202. PUT Resource accountCodeStructrue

REST resource elements	Details
HTTP method	PUT
Resource URI	`https://serverName:port/tuamConsole/rest/accountCodeStructrues`
URI query parameters	None
Request headers	Content-Type, Accept

Table 202. PUT Resource accountCodeStructrue (continued)

REST resource elements	Details
Request payload	**Response XML** `<ns:accountCodeStructure xmlns:ns=http://www.ibm.com/xmlns/prod/TUAM/1.0` `accountStructureName="STANDARD">` `<startPosition>1</startPosition>` `<accountLevels>` `<accountLevel>` `<accountLevelOrder>1</accountLevelOrder>` `<accountLength>16</accountLength>` `<accountDescription>accountDescription</accountDescription>` `</accountLevel>` `<accountLevel>` `<accountLevelOrder>accountLevelOrder</accountLevelOrder>` `<accountLength>accountLength</accountLength>` `<accountDescription>accountDescription</accountDescription>` `</accountLevel>` `</accountLevels>` `</ns:accountCodeStructure>` **Response JSON** `{` `"accountLevels": [` `{` `"accountDescription": "level1",` `"accountLength": 16,` `"accountLevelOrder": 1` `},` `{` `"accountDescription": "level2",` `"accountLength": 16,` `"accountLevelOrder": 2` `}` `],` `"accountStructureName": "STANDARD",` `"startPosition": 0` `}`
Request Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue` `orapplication/vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue`
Response headers	Content-Type

Table 202. PUT Resource accountCodeStructrue (continued)

REST resource elements	Details
Response payload	**Response XML** `<ns:accountCodeStructure xmlns:ns=http://www.ibm.com/xmlns /prod/TUAM/1.0 accountStructureName="STANDARD"> <startPosition>1</startPosition> <accountLevels> <accountLevel> <accountLevelOrder>1</accountLevelOrder> <accountLength>16</accountLength> <accountDescription>accountDescription </accountDescription> </accountLevel> <accountLevel> <accountLevelOrder>accountLevelOrder </accountLevelOrder> <accountLength>accountLength</accountLength> <accountDescription>accountDescription </accountDescription> </accountLevel> </accountLevels> <uri>...../accountCodeStructures/STANDARD</uri> </ns:accountCodeStructure>`  **Response JSON** `{ "accountLevels": [{ "accountDescription": "level1", "accountLength": 16, "accountLevelOrder": 1 }, { "accountDescription": "level2", "accountLength": 16, "accountLevelOrder": 2 }], "accountStructureName": "STANDARD", "startPosition": 0, "uri": "...../accountCodeStructures/STANDARD" }`
Response Content-Type	`application/vnd.ibm.TUAM+xml;version=1;format=accountCodeStructrue` `orapplication/ vnd.ibm.TUAM+json;version=1;format=accountCodeStructrue`
Generic operation	None
Normal HTTP response codes	**200**: The `accountCodeStructrue` has been successfully modified. The new `accountCodeStructrue` is included in the response.

Table 202. PUT Resource accountCodeStructrue (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned in the following cases: • if the `accountStructureName` does not exist in the request. • if the start position of the `accountCodeStructure` is less than one or greater that the max length. • if no level is specified. • if any account level does not have a description. • if the full length of the account code structure exceeds the maximum length. **403**: This code is returned if the requester does not have permission to update the specified user. **404**: This code is returned if the requested `accountCodeStructrue` is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

DELETE Resource accountCodeStructrues

Use the DELETE Resource `accountCodeStructrues` to delete the representation of an existing SmartCloud Cost Management `accountCodeStructrue` for a given `accountStructrueName`.

Table 203. DELETE Resource accountCodeStructrues

REST resource elements	Details
HTTP method	DELETE
Resource URI	`https://serverName:port/tuamConsole/rest/accountCodeStructrues/{id}`
URI query parameters	None
Request headers	None
Request payload	None
Request Content-Type	None
Response headers	None
Response payload	None
Response Content-Type	None
Generic operation	None
Normal HTTP response codes	**204**: The `accountCodeStructrue` is deleted as requested.

Table 203. DELETE Resource accountCodeStructrues (continued)

REST resource elements	Details
Error HTTP response codes	**400**: This code is returned if the `accountStructrueName` is not passed in the request. **403**: This code is returned if the requester does not have permission to delete the user. **404**: This code is returned if the requested `accountCodeStructrue` is not defined. **500**: This code is returned if the SmartCloud Cost Management application encountered an internal error while processing the request.

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 SmartCloud Cost Management 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`.

The following table 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 email message. You can also use these attributes to send a separate email message for each individual job. These attributes have default values. If you do not include these attributes or provide blank values, the default values are used.

Table 204. Jobs Element Attributes

Attribute	Required or Optional	Description
`processFolder`	Optional	In most cases, you will not need to use this attribute. By default, the path to the `processes` directory set in the SmartCloud Cost Management ConfigOptions table is used. If you set this attribute, it will override the path to the`processes` directory that is set in the ConfigOptions table. **Example of the use of this attribute:** The `processFolder` attribute can be used to allow job processing to occur on a SmartCloud Cost Management application server that is not usually used for processing. For example, assume you have a single database server and process most of your feeds on a UNIX or Linux SmartCloud Cost Management application server. However, there are some feeds that you process on another SmartCloud Cost Management application server. You can configure the job file path in the database to point to the processes directory on the main server. On the other server, you can set the processFolder attribute in the job file to point to the processes path on that server. The result is that both SmartCloud Cost Management servers can use a single database, but process data on more than one server.
`smtpSendJobLog`	Optional	Specifies whether the job log should be sent via email. Valid values are: • `true`" (send via email) • `false`" (do not send) The default is "`false`".
`smtpServer`	Optional	The name of the SMTP mail server that will be used to send the job log. The default is "`mail.ITUAMCustomerCompany.com`".
`smtpFrom`	Optional	The fully qualified email address of the email sender. The default is "`ITUAM@ITUAMCustomerCompany.com`".

Table 204. Jobs Element Attributes (continued)

Attribute	Required or Optional	Description
`smtpTo`	Optional	The fully qualified email address of the email 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` • servername:userID:password:port: user@domain Example: `mail.xyzco.com:janes:global:25: jsmith@xyzco.com` If you want to use multiple addresses, separate them with a comma (,). 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`"
`smtpSubject`	Optional	The text that you want to appear in the email subject. The default subject is: `ITUAM job <job name> running on <server name> completed <successfully or with x warning(s)/with x error(s)>`
`smtpBody`	Optional	The text that you want to appear in the email body. The default body text is: `Attached are results from a JobRunner execution.`

Job Element

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.

The following table 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 205. Job Element Attributes

Attribute	Required or Optional	Description
id	Required	A text string name for the job. This value must be unique from other job ID values in the file. **Example** `id="Nightly"` In this example, the subfolder that contains log files for this job will also be named Nightly.
description	Optional	A text string description of the job (maximum of 255 characters). **Example** `description="Nightly collection and processing"`
active	Optional	Specifies whether the job should be run. Valid values are: • true" (run the job) • false" (do not run the job) The default is "true".
dataSourceId	Optional	The data source for the SmartCloud Cost Management database. **Example** `dataSourceId=ITUAMDev` If this parameter is not provided, the data source that is set as the default processing data source on Administration Console Data Source List Maintenance page is used. To use a data source other than the default, set this parameter to the appropriate data source ID
joblogShowStepParameters	Optional	Specifies whether parameters for the steps in a job 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".
joblogShowStepOutput	Optional	Specifies whether output generated by the steps in a job 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".

Table 205. Job Element Attributes (continued)

Attribute	Required or Optional	Description
processFolder	Optional	In most cases, you will not need to use this attribute. By default, the path to the processes directory set in the SmartCloud Cost Management ConfigOptions table is used. If you set this attribute, it will override the path to theprocesses directory that is set in the ConfigOptions table. **Example of the use of this attribute:** The processFolder attribute can be used to allow job processing to occur on a SmartCloud Cost Management application server that is not usually used for processing. For example, assume you have a single database server and process most of your feeds on a UNIX or Linux SmartCloud Cost Management application server. However, there are some feeds that you process on a Windows SmartCloud Cost Management application server. You can configure the job file path in the database to point to theprocesses directory on the UNIX server. On the Windows server, you can set the processFolder attribute in the job file to point to the processes path on the Windows server. The result is that both SmartCloud Cost Management servers can use a single database, but process data on more than one server platform.
processPriorityClass	Optional	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 Software Support 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.
joblogWriteToTextFile	Optional	Specifies whether the job log should be written to a text file. Valid values are: • true" (writes to a text file) • false" (does not write to a text file) The default is "true".
joblogWriteToXMLFile	Optional	Specifies whether the job log should be written to an XML file. Valid values are: • true" (writes to an XML file) • false" (does not write to an XML file) The default is "true".
smtpSendJobLog	Optional	Specifies whether the job log should be sent via email. Valid values are: • true" (send via email) • false" (do not send) The default is "false".
smtpServer	Optional	The name of the SMTP mail server that will be used to send the job log. The default is "mail.ITUAMCustomerCompany.com".
smtpFrom	Optional	The fully qualified email address of the email sender. The default is "ITUAM@ITUAMCustomerCompany.com".

Table 205. Job Element Attributes (continued)

Attribute	Required or Optional	Description
smtpTo	Optional	The fully qualified email address of the email receiver.
smtpSubject	Optional	The text that you want to appear in the email subject. The default subject is: ITUAM job <job name> running on <server name> completed <successfully or with x warning(s)/with x error(s)>
smtpBody	Optional	The text that you want to appear in the email body. The default body text is: Attached are results from a JobRunner execution.
stopOnProcessFailure	Optional	Specifies whether a job with multiple processes should stop if any of the processes fail. Valid values are: • true" (stop processing) • false" (continue processing) The default is "false". **Note:** If stopOnStepFailure is set to "false"at the Steps element level in a process, processing continues regardless of the value set for stopOnProcessFailure.

Process element

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 file system, for example).

The following table 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 206. Process Element Attributes

Attribute	Required or Optional	Description
id	Required	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. If the buildProcessFolder attribute is not included or is set to "true" (the default), 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="ABCSoftware" In this example, the process definition folder created by Job Runner will be named ABCSoftware.
description	Optional	A text string description of the process (maximum of 255 characters). **Example** description="Process for ABCSoftware"

Table 206. Process Element Attributes (continued)

Attribute	Required or Optional	Description
`buildProcessFolder`	Optional	Specifies whether Job Runner will create a process definition folder with the same name as the `id` attribute value in the `processes` folder. 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 Job Runner to run a script or program that does not require a process definition folder. 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"`.
`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 Software Support 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"`.

Steps Element

A `Steps` element is a container for one or more `Step` elements. The `Steps` element has one optional attribute.

Table 207. Steps Element Attribute

Attribute	Required or Optional	Description
`stopOnStepFailure`	Optional	Specifies whether processing should continue if any of the active steps in the process fail. Valid values are: • `true`" (processing fails) If the `stopOnProcessFailure` attribute is also set to "`true`", the remaining processes in the job are not executed. If `stopOnProcessFailure` is set to "`false`", the remaining processes in the job are executed. • `false`" (processing continues) In this situation, all remaining processes in the job are also executed regardless of the value set for `stopOnProcessFailure`. The default is "`true`".

Step Element

A `Step` element defines a step within a process.

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

The following table 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 208. Step Element Attributes

Attribute	Required or Optional	Description
`id`	Required	A text string name for the step. This value must be unique from other step ID values in the process. **Example** `id="Scan"` In this example, the step is executing the Scan program.
`description`	Optional	A text string description of the step (maximum of 255 characters). **Example** `description="Scan ABCSoftware"`

Table 208. Step Element Attributes (continued)

Attribute	Required or Optional	Description
`active`	Optional	Specifies whether the step should be run. Valid values are: • `true`" (run the step) • `false`" (do not run the step) The default is "`true`".
`type`	Required	The type of step that is being implemented: `"ConvertToCSR"` or `"Process"`. `ConvertToCSR` specifies that the step performs data collection and conversion and creates a CSR file. `Process` specifies that the step executes a program such as Scan, Acct, Bill, and so forth.
`programName`	Required	The name of the program that will be run by the step. It the program name is any of the following, the `programType` attribute must be `java`: • Integrator • SendMail • Acct • Bill • Sort • DBLoad • DBPurge • JobFileConversion • Rpd • Scan • Cleanup • FileTransfer • WaitFile

Table 208. Step Element Attributes (continued)

Attribute	Required or Optional	Description
		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, Job Runner searches the `collectors`, `bin`, and `lib` directories for the program in the order presented. If the `type` attribute is `Process`, this value is the name of a SmartCloud Cost Management program (for example, `"Scan"`, `"Acct"`,`"Bill"`, `"DBLoad"`, and so forth). Examples: programName="WinDisk.exe" `programName="Cleanup"`
`processPriorityClass`	Optional	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 Software Support 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.
`programType`	Optional	The type of program specified by the `programName` attribute: • `"console"`-Console Application • `"com"`-COM Component (deprecated, compatible with earlier versions only) • `"net"`-.Net Component • `"java"`-Java application

Table 208. Step Element Attributes (continued)

Attribute	Required or Optional	Description
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".
joblogShowStepOutput	Optional	Specifies whether output generated by the step 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".

Parameters Element

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

Parameter element

A Parameter element defines a parameter to a step.

The valid attributes for collection 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.

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 Acct and Bill, 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 Acct and the proration table used by Integrator. You can place these files in a central location so that they can be used by multiple processes. In this case, you must provide the path.
- Attributes include macro capability so that the following predefined 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.
  - **%CollectorLogs%**. Specifies the collector log files folder as defined as in the CIMSConfigOptions table.
  - **%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.

– **%<*Date Keyword*>_Start%**. For files that contain a date in the file name, specifies that files with dates matching the first day of the <*Date Keyword*> parameter value are used. For example, if the <*Date Keyword*> 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.
– **%<*Date Keyword*>_End%**. For files that contain a date in the file name, specifies that files with dates matching the last day of the <*Date Keyword*> parameter value are used. For example, if the <*Date Keyword*> 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 in the file name, specifies that files with dates 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.
– **%LogDate_End%.** For files that contain a date in the file name, specifies that files with dates 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.

The valid attributes for process step parameters (type=Process) are listed in the following sections. The attributes are broken down as follows

- Parameter attributes that are specific to a program (Scan, Acct, Bill, etc.).
- Parameter attributes that are specific to a program type (wsf, com, net, console, etc.).

Acct specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the acct program.

Attributes

Table 209. Acct specific parameter attributes

Attribute	Required or Optional	Description
accCodeConvTable	Optional	The name of account code conversion table used by Acct. Include a path if the table is in a location other than the collector's process definition directory. **Examples:** `accCodeConvTable="MyAcctTbl.txt"` `accCodeConvTable="E:\Processes\Account\ MyAcctTbl.txt` The default is "AcctTabl.txt".

Table 209. Acct specific parameter attributes (continued)

Attribute	Required or Optional	Description
controlCard	Optional	A valid Acct control statement or statements. All Acct control statements are stored in the Acct control file. **Note:** If you have an existing Acct control file in the process definition directory, the statements that you define as controlCard parameters will overwrite all statements currently in the file. To define multiple control statements, use a separate parameter for each statement. **Example** <Parameter controlCard="TEST A"/ <Parameter controlCard="VERIFY DATA ON"/>
controlFile	Optional	The name of the control file used by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** controlFile="MyAcctCntl.txt" The default is "AcctCntl.txt".
exceptionFile	Optional	The name of the exception file produced by Acct. This file must be in the collector's process definition directory-do not include a path. The file name should contain the log date so that it is not overwritten when Acct is run again. **Example** exceptionFile= "Exception_%LogDate_End%.txt" The default is "Exception.txt".
inputFile		The name of CSR or CSR+ file to be processed by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** inputFile="MyCSR.txt" The default is "CurrentCSR.txt".
outputFile		The name of the output CSR or CSR+ file produced by Acct. This file must be in the collector's process definition directory-do not include a path. **Example** outputFile="CSR.txt" The default is "AcctCSR.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example account code conversion traces) • "false" (Less detailed information is provided) The default is "false".

Bill specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the bill program.

Attributes

Table 210. Bill specific parameter attributes

Attribute	Required or Optional	Description
`controlCard`	Optional	A valid Bill control statement or statements. All Bill control statements are stored in the Bill control file. **Note:** If you have an existing Bill control file in the process definition directory, the statements that you define as `controlCard` parameters will overwrite all statements currently in the file. To define multiple control statements, use a separate parameter for each statement. **Example** `<Parameter controlCard="CLIENT SEARCH ON"/>` `<Parameter controlCard="DEFINE J1 1 1"/>`
`controlFile`	Optional	The name of the control file used by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `controlFile="MyBillCntl.txt"` The default is `"BillCntl.txt"`.
`detailFile`	Optional	The name of the Detail file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `detailFile=` `"MyDetail.txt"` The default is `"BillDetail.txt"`.
`dateSelection`	Optional	Defines a date range for records to be processed by Bill. Valid values are a from and to date range in yyyymmdd format or a date keyword. **Examples** `dateSelection="2008117 2008118"` In this example, Bill will process records with an accounting end dates of January 17 and 18, 2007. `dateSelection="PREDAY"` In this example, Bill will process records with an accounting end date one day prior to the date Job Runner is run.
`defaultRateTable`	Optional	Defines the default Rate Table to be used when matching a Resource Entry to a Rate. The `'Standard'` Rate table is used if this option is not specified.
`identFile`	Optional	The name of the Ident file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `identFile="MyIdent.txt"` The default is `"Ident.txt"`.

Table 210. Bill specific parameter attributes (continued)

Attribute	Required or Optional	Description
inputFile		The name of the CSR or CSR+ file to be processed by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `inputFile="CSR.txt"` The default is "AcctCSR.txt".
keepZeroValueResources	Optional	This parameter when set to true, enables resources with zero values to be written to or read from CSR files and in billing output. Resources with zero values are normally discarded. The default is "false".
multTableFile	Optional	The name of the proration table used by Prorate. Include a path if the table is in a location other than the collector's process definition directory. **Examples** `multTableFile="MyMultTable.txt"` `multTableFile="E:\Processes\Prorate\MyMultTable.txt"`
rateSelection	Optional	Defines the algorithm used to determine what Historical Rate over an Accounting period should be used when matching a Resource Entry to a Rate. Valid options are NONE, FIRST, and LAST. The default value is NONE. • NONE: If only one rate is effective over the Accounting period, that rate is used. If more than one rate is effective over the Accounting period, no Rate is matched. • FIRST: The first rate effective over the Accounting period is used. • LAST: The last rate effective over the Accounting period is used.
reportDate	Optional	Defines the dates that are used as the accounting start and end dates in the Summary records created by Bill. Valid values are a date in yyyymmdd format or a 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 "CURMON", which sets the accounting start date to the first of the month and the end date to the last day of the month.
resourceFile		The name of the Resource file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `resourceFile="MyResource.txt"` There is no default. This file is not produced if this attribute is not provided.
summaryFile	Optional	The name of the Summary file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** `summaryFile="MySummary.txt"` The default is "BillSummary.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example accounting date calculation tracing and client search tracing) • "false" (Less detailed information is provided) The default is "false".

Cleanup specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the Cleanup program.

Attributes

Table 211. Cleanup specific parameter attributes

Attribute	Required or Optional	Description
`dateToRetainFiles`	Optional	A date by which all yyyymmdd files that were created prior to this date will be deleted. You can use a date keyword or the date in yyyymmdd format. **Example** `dateToRetainFiles="PREMON"` This example specifies that all files that were created prior to the previous month will be deleted.
`daysToRetainFiles`		The number of days that you want to keep the yyyymmdd files after their creation date. **Example** `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.
`cleanSubfolders`	Optional	Specifies whether the files that are contained in subdirectories are deleted. Valid values are: • `"true"` (the files are deleted) • `"false"` (the files are not deleted) The default is `"false"`.
`folder`	Optional	By default, the Cleanup program deletes files with file names containing the date in yyyymmdd format from the collector's process definition directory. If you want to delete files from another directory, use this attribute to specify the path and directory name. **Example** `folder="\\Server1\LogFiles`

Console parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the CONSOLE program type.

Attributes

Table 212. Console parameter attributes

Attribute	Required or Optional	Description
`scanFile`	Optional	This attribute is applicable only if the Smart Scan feature is enabled . 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. 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` 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.

Console specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the CONSOLE program type.

Attributes

Table 213. Console specific parameter attributes

Attribute	Required or Optional	Description
`useCommandProcessor`	Optional	Specifies whether the `Cmd.exe` program should be used to execute a console program. If the `Cmd.exe` program is not used, then the console program is called using APIs. Valid values are: • `"true"` (the `Cmd.exe` program is used) • `"false"` (the `Cmd.exe` program is not used) The default is `"true"`.

Table 213. Console specific parameter attributes (continued)

Attribute	Required or Optional	Description
useStandardParameters		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: • LogDate • RetentionFlag • Feed • OutputFolder These parameters are passed before any other parameters defined for the step. Valid values are: • "true" (the standard parameters are passed) • "false" (the standard parameters are not passed) If the step type is Process, the default value is "false". If the step type is ConvertToCSR, the default is "true".
XMLFileName, CollectorName, and CollectorInstance	Optional	These attributes are used by the Windows Disk and Windows Event Log collectors. They specify the name of the XML file used by the collector; the name of the collector; and the collector instance, respectively.

DBLoad specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the DBLoad program.

Attributes

Table 214. DBLoad specific parameter attributes

Attribute	Required or Optional	Description
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".
bypassDetailDuplicateCheck	Optional	Allows the user to skip the Detail Duplicate Check. although the Summary Duplicate Check is still performed. It is recommended that this duplicate check is performed. However, you may want to skip this check to avoid unnecessary overhead or to increase the speed of load. Valid values are: • "true" (bypass detail duplicate check) • "false" (perform detail duplicate check) The default is "false".

Table 214. DBLoad specific parameter attributes (continued)

Attribute	Required or Optional	Description
bypassDuplicateCheck	Optional	Allows the user to skip the Detail and Summary Duplicate Check. It is recommended that these duplicate checks are performed. However, you may want to skip these checks to avoid unnecessary overhead or to increase the speed of load. Valid values are: • "true" (bypass detail and summary duplicate checks) • "false" (perform detail and summary duplicate checks) The default is "false".
detailFile	Optional	The name of the Detail file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. detailFile= "MyDetail.txt" The default is "BillDetail.txt".
identFile	Optional	The name of the Ident file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** identFile="MyIdent.txt" The default is "Ident.txt".
loadType		By default, the DBLoad program loads the Summary, Detail, Ident, and Resource (optional) files into the database. If you want to load a specific file rather than all files, the valid values are: • Summary • Detail • Resource • Ident • DetailIdent (loads Detail and Ident files) • All (loads Summary, Detail, and Ident file types)
resourceFile		The name of the Resource file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** resourceFile="MyResource.txt" There is no default. This file is not produced if this attribute is not provided.
summaryFile	Optional	The name of the Summary file that is produced by Bill. This file must be in the collector's process definition directory-do not include a path. **Example** summaryFile="MySummary.txt" The default is "BillSummary.txt".
trace		Specifies the level of processing detail that is provided in trace files. The more detailed the message level, the more quickly the trace file might reach the maximum size. • "true" (More detailed information is provided, for example accounting date calculation tracing and client search tracing) • "false" (Less detailed information is provided) The default is "false".

Table 214. DBLoad specific parameter attributes (continued)

Attribute	Required or Optional	Description
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".

DBPurge specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the DBPurge program.

Attributes

Note: For an example of the use of DBPure in a job file, see the SampleDBPurge.xml file in <*SCCM_install_dir*>\samples\jobfiles\.

Table 215. DBPurge specific parameter attributes

Attribute	Required or Optional	Description
MonthsToKeep	Optional	Specifies the age of the loads (in months) that you want to delete from the tables.
PurgeSummary, PurgeBillDetail, PurgeIdent, and PurgeAcctDetail.	Required	Specify whether the CIMSSummary, CIMSDetail, CIMSDetailIdent, or CIMSResourceUtilization tables are purged. Valid values are: • "true" (The table is purged) • "false" (The table is not purged)
StartDate and EndDate **Note:** MonthsToKeep parameter and the StartDate and StopDate parameters are mutually exclusive. If all parameters are specified, StartDate and StopDate parameters are ignored.	Optional	Specifies the date range for the loads that you want to delete from the tables. Any loads that have accounting start and end dates in this range are deleted. Valid values are: • preday (previous day) • rndate (current day) • premon (previous month) • curmon (current month) • date in yyyymmdd format If you use the premon or curmon keyword for the start date or end date, the first day of the month is used for the start date and the last day of the month is used for the end date.

FileTransfer specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the FileTransfer program.

Parameters

Table 216. FileTransfer specific parameter attributes

Attribute	Required or Optional	Description
`continueOnError`		For a multi-file transfer, specifies whether subsequent file transfers continue if a transfer fails. Valid values are: • `"true"` (file transfer continues) • `"false"` (file transfer does not continue) The default is `"false"`.
`pollingInterval`		The number of seconds to check for file availability (maximum of 10,080 [one week]). **Example** `pollingInterval="60"` This example specifies a polling interval of 60 seconds. The default is 5 seconds.
`type`	Required	The type of file transfer. Valid values are: • `"ftp"` (File Transfer Protocol [FTP] transfer) • "file" (Local transfer) • "ssh" (Secure Shell transfer)
`UseSFTP`	Optional	It the `type` attribute value is `"ssh"`, this attribute specifies whether the SFTP or SCP protocol will be used for transferring files. Valid values are: • `"true"` (the SFTP protocol is used) • `"false"` (the SCP protocol is used) The default is `"false"`. A value of `"true"` is used with certain SSH servers (such as Tectia SSH Server 5.x) to allow file transfers to complete successfully.

The following attributes `from`, `to`, `action`, and `overwrite` are attributes of a single `Parameter` element. If you are transferring multiple files, include a `Parameter` element with these attributes for each file. For an example of these attributes in a job file, see the `SampleNightly.xml` file.

Table 217. FileTransfer parameters

Attribute	Required or Optional	Description
`action`	Required	Specifies the file activity. Valid values are: • `"Copy"` (copies the file from the `from` location to the `to` location) • `"Delete"` (deletes the file from the `from` location) • `"Move"` (copies the file from the `from` location to the `to` location and then deletes the file from the `from` location) The default is `Copy`.

Table 217. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
from and to	Required	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: • type="ftp" The "ftp://" file transfer protocol can be used for either the from or the to location. However, because the job file must be run from the SmartCloud Cost Management application server system, the file transfer samples show the "ftp://" file transfer protocol being used only in the from parameter. Note the following when using the from and to attributes with type="ftp": – Values must be specified for serverName, userId, and userPassword parameters. – The transferType attribute is optional. Valid values are binary (default) or ascii. – The from and to parameters can include SmartCloud Cost Management macros (for example, %LogDate_End%). – The from location can include wildcards in the file name. If the from parameter includes file name wildcards, the to parameter must specify a directory name (no file names). For an example of the use of the from and to parameters for an FTP transfer, refer to the sample job file SampleFileTransferFTP_withPW.xml.
		• type="file" The "file://" file transfer protocol is used for both the from and to parameters. Note the following when using the from and to attributes with type="file" – The serverName, userId, and userPassword parameters are not required. – The from location can be a local path, a Windows UNC path, a Windows mapped drive, or a UNIX mounted drive path. – The from location must be a path that is local to the from location. On Windows, this can include a local hard disk drive, a Windows UNC path, or a Windows mapped drive. On UNIX, this can include a local directory or a UNIX mounted drive path – The from and to parameters can include SmartCloud Cost Management macros (for example, %LogDate_End%). – The from location can include wildcards in the file name. If the from parameter includes file name wildcards, the to parameter must specify a directory name (no file names). For an example of the use of the from and to parameters for a Windows transfer, refer to the sample job file SampleFileTransferLocal_noPW.xml.

Table 217. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
		• `type="ssh"` The `"ssh://"` file transfer protocol can be used for either the `from` or the `to` location. However, because the job file must be run from the SmartCloud Cost Management application server system, the file transfer samples show the `"ssh://"` file transfer protocol being used only in the `from` parameter. Note the following when using the `from` and `to` attributes with `type="ssh"` – Values must be specified for `serverName`, `userId`, and `userPassword` parameters. – If you are using a keyfile to authenticate with the the ssh daemon (sshd on UNIX or Linux), refer the information in the "Sample SSH file transfer job file" section of *Administering data processing*. - The user ID on the remote system must be configured to trust the SmartCloud Cost Management application server user ID. This means that if the keyfile path is specified, the public key for the SmartCloud Cost Management application server user ID must be in the authorized keys file of the remote user. – The `from` and `to` parameters must use the UNIX-style naming convention, which uses forward slashes (for example, `ssh:///anInstalledCollectorDir/CollectorData.txt`), regardless of whether you are transferring files from a UNIX or Linux system or a Microsoft Windows system. – You can use the ssh file transfer protocol on UNIX or Linux systems. However, before you run a job file that uses the ssh protocol, verify that you can successfully issue ssh commands from the SmartCloud Cost Management application server to the remote ssh system. – The `from` and `to` parameters can include SmartCloud Cost Management macros (for example, `%LogDate_End%`). – The `from` location can include wildcards in the file name. If the `from` parameter includes file name wildcards, the `to` parameter must specify a directory name (no file names).
`overwrite`	Optional	Specifies whether the destination file is overwritten. Valid values are: • `"true"` (the file is overwritten) • `"false"` (the file is not overwritten) The default is `"false"`.

The following attributes are for FTP transfer only.

Table 218. FileTransfer parameters

Attribute	Required or Optional	Description
`connectionType`	Optional	Describes how the connection address is resolved. This is an advanced configuration option that should be used only after consulting IBM Software Support. Valid values are: • `"PRECONFIG"` (retrieves the proxy or direct configuration from the registry) • `"DIRECT"` (resolves all host names locally) • `"NOAUTOPROXY"` (retrieves the proxy or direct configuration from the registry and prevents the use of a startup Microsoft JScript or Internet Setup (INS) file) • `"PROXY"` (passes requests to the proxy unless a proxy bypass list is supplied and the name to be resolved bypasses the proxy) The default is `"PRECONFIG"`.

Table 218. FileTransfer parameters (continued)

Attribute	Required or Optional	Description
passive	Optional	Forces the use of FTP passive semantics. In passive mode FTP, the client initiates both connections to the server. This solves the problem of firewalls filtering the incoming data port connection to the FTP client from the FTP server. This is an advanced configuration option that should be used only after consulting IBM Software Support.
proxyServerBypass	Optional	This is a pointer to a null-terminated string that specifies an optional comma-separated list of host names, IP addresses, or both, that should not be routed through the proxy. The list can contain wildcards. This options is used only when connectionType="PROXY". This is an advanced configuration option that should be used only after consulting IBM Software Support.
proxyServer	Optional	If connectionType="PROXY", the name of the proxy server(s) to use. This is an advanced configuration option that should be used only after consulting IBM Software Support.
serverName	Required	A valid FTP IP address or server name. Example: serverName="ftp.xyzco.com"
transferType	Optional	The type of file transfer. Valid values are: • "binary" • "ascii" The default is "binary".
IsZOS	Optional	Indicates whether the FTP connection is to a Z/OS file system or not. Valid values are: • "true" • "false" The default value is "false"
userId	Optional	The user ID used to log on to the FTP server.
userPassword	Optional	The user password used to log on to the FTP server.

Rebill specific parameter attributes

This topic outlines all the possible settings that can be entered using the parameter element in the Rebill program.

Attributes

Table 219. Rebill specific parameter attributes

Attribute	Required or Optional	Description
DataSourceName	Optional	This allows you to set the data source to use for the ReBill operation. Default is the processing data source.
EndDate	Required	The end date (YYYYMMDD) of the range in the CIMSSummary table that will be converted. Can also pass keywords such as CURDAY, PREDAY, RNDATE, PREMON or CURMON. If PREMON or CURMON are passed, it will use a range for StartDate/EndDate.
RateCode	Required	The rate code that should be converted. Default is all.

Table 219. Rebill specific parameter attributes (continued)

Attribute	Required or Optional	Description
`RateTable`	Optional	The rate table for the rate codes that should be converted. For the standard rate table this would be: `Parameter RateTable="STANDARD"` .
`StartDate`	Required	The start date (YYYYMMDD) of the range in the CIMSSummary table that will be converted. Can also pass keywords such as `CURDAY`, `PREDAY`, `RNDATE`, `PREMON` or `CURMON`. If `PREMON` or `CURMON` are passed, it will use a range for `StartDate/EndDate`.

Scan specific parameter attributes

This topic outlines all the possible attributes that can be set using the parameter element in the scan program.

Attributes

Table 220. Scan specific parameter attributes

Header	Header	Header
`allowEmptyFiles`	Optional	Specifies whether a warning or error occurs when feed subdirectories contain a zero-length file that matches the log date value. Valid values are: • `"true"` (a warning occurs, processing continues) • `"false"` (an error occurs, processing fails) The default is `"false"`.
`allowMissingFiles`	Optional	Specifies whether a warning or error occurs when feed subdirectories do not contain a file that matches the log date value. Valid values are: • `"true"` (a warning occurs, processing continues) • `"false"` (an error occurs, processing fails) The default is `"false"`.
`excludeFile`	Optional	The name of a file to be excluded from the Scan process. The file can be in any feed subdirectory in the collector's process definition folder. The file name can include wildcard characters but not a path. **Example** `excludeFile="MyCSR*"` In this example, all files that begin with `MyCSR` are not scanned.
`excludeFolder`	Optional	The name of a feed subfolder to be excluded from the Scan process. The subdirectory name can include wildcard characters but not a path. The feed directory must be a top-level directory within the process definition folder. **Example** `excludeFolder="Server1"` In this example, the feed subdirectory `Server1` is not scanned.

Table 220. Scan specific parameter attributes (continued)

Header	Header	Header
includeFile	Optional	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 subdirectory in collector's process definition directory. **Example** `includeFile="MyCSR.txt"` In this example, files in the feed subdirectorys that are named MyCSR are scanned.
retainFileDate	Optional	Specifies whether the date is retained in the final CSR file (i.e., yyyymmdd.txt rather than CurrentCSR.txt). Valid values are: • "true" (the file name is yyyymmdd.txt) • "false" (the file name is CurrentCSR.txt) The default is "false".
useStepFiles	Optional	Specifies whether the Smart Scan feature is enabled. 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 subdirectories (for example, CIMSWinProcess/Server1/20080624.txt). If you want to override the default name, use the parameter attribute scanFile in the collection step.

WaitFile specific parameter attributes

This topic outlines all the possible attributes that can be entered using the parameter element in the WaitFile program.

Attributes

Table 221. WaitFile specific parameter attributes

Attribute	Required or Optional	Description
pollingInterval		The number of seconds to check for file availability (maximum of 10,080 [one week]). **Example** `pollingInterval="60"` This example specifies a polling interval of 60 seconds. The default is 5 seconds.
timeout	Optional	The number of seconds that Job Runner will wait for the file to become available. If the timeout expires before the file is available, the step fails. **Example** `timeout="18000"` This example specifies a timeout of 5 hours. The default is to wait indefinitely.

Table 221. WaitFile specific parameter attributes (continued)

Attribute	Required or Optional	Description
timeoutDateTime	Optional	A date and time up to which 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. **Example** `timeoutDateTime="%rndate% 23:59:59"` This example specifies a timeout of 23:59:59 on the day Job Runner is run. The default is to wait indefinitely.
filename	Required	The name of the file to wait for. If a path is not specified, the path to the process definition directory 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="BillSummary_` `%LogDate_End%.txt"` In this example, Job Runner will wait for Summary files that contain the same end date as the %LogDate_End% value.

Defaults Element

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

Default Element

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.

There are two types of attributes that you can use in a Default element: pre-defined and user defined as shown in the following table.

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 222. Default Element Attributes

Attribute	Description
Pre-defined attributes. These are the attributes that are pre-defined for SmartCloud Cost Management.	
LogDate	The log date specifies the date for the data that you want to collect. You should enter the log date in the job file only if you are running a snapshot collector or the Transactions collector.
RetentionFlag	This attribute is for future use. Valid values are KEEP or DISCARD.

Table 222. Default Element Attributes (continued)

Attribute	Description
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.

Default Example

This job file example contains two Default elements.

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

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

```
<?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">
   <Defaults>
   <Default programName="CIMSACCT"
   accCodeConvTable="C:\ITUAM\AccountCodeTable\AccountCodeTable\
   AcctTabl-Win.txt"/>
   </Defaults>
   <Process   id="DBSpace"
   description="Process for DBSpace Collection"
    active="true">
   <Defaults>
   <Default LogDate="RNDATE"/>
   </Defaults>
   <Steps>
   :
   :
```

Integrator job file structure

The Integrator provides a flexible means of modifying the input data that you collect from usage metering files. Integrator processes the data in an input file according to stages that are defined in the job file XML.

Each stage 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. You can add, remove, or activate, stages as needed.

Note: In addition to usage metering files, you can also use integrator to collect data from a variety of other files, including CSR and CSR+ files.

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

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 identifies the type of file to be processed.

Example

In the following example, the input file is a CSR file.

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

Where:

- The `Input name` value can be:
  - **`"CSRInput"`**

    This value is used to parse a CSR or CSR+ type of file. An example of this value is provided in most sample collector job files.
  - **`"AIXAAInput"`**

This value is used to parse data in an Advanced Accounting log file. For an example of the use of this value, see the SampleAIXAA.xml job file.

– **"ApacheCommonLogFormat"**

  This value is used to parse data in an Apache log where the records are in the Apache Common Log Format (CLF). For an example of the use of this value, see the SampleApache.xml job file.

– **"DBSpaceInput"**

  This value is used to parse data gathered directly from a SQL Server or Sybase database or databases. For an example of the use of this value, see the SampleDBSpace.xml job file.

– **"MSExchange2003"**

  This value is used to parse data in a Microsoft Exchange Server 2000 or 2003 log file that records the activities on the server. For an example of the use of this value, see the SampleExchangeLog.xml job file.

– **"NCSAInput"**

  This value is used to parse data in a WebSphere HTTP Server access log. For an example of the use of this value, see the SampleWebSphereHTTP.xml job file.

– **"NotesDatabaseSizeInput"**

  This value is used to parse data in a Notes Catalog database (catalog.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

– **"NotesEmailInput"**

  This value is used to parse data in a Notes Log Analysis database (loga4.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

– **"NotesUsageInput"**

  This value is used to parse data in a Notes Log database (catalog.nsf). For an example of the use of this value, see the SampleNotes.xml. job file.

– **"SAPInput"**

  This value is used to parse data in an SAP log file.

– **"SAPST03NInput"**

  This value is used to parse a data in an SAP Transaction Profile report.

– **"VmstatInput"**

  This value is used to parse data in a vmstat log file. For an example of the use of this value, see the SampleVmstat.xml job file.

– **"SarInput"**

  This value is used to parse data in a sar log file. For an example of the use of this value, see the SampleSar.xml job file.

– **"W3CWinLog"**

  This value is used to parse data in a Microsoft Internet Information Services (IIS) log file. For an example of the use of this value, see the SampleMSIIS.xml job file.

– **"HPVMSarInput"**

  This value is used to parse data in a HP VM sar log file. For an example of the use of this value, see the SampleHPVMSar.xml job file.

– **"KVMInput"**

  This value is used to parse data in a KVM log file. For an example of the use of this value, see the SampleKVM.xml job file.

– **"CollectorInput"**

This value is followed by one of the following `Collector name` attributes:

- **DATABASE, DELIMITED, FIXEDFIELD, or REGEX**

  These value are used to

  - Collect data from a database.
  - Parse data in a file that contains delimited record fields.
  - Parse data in a file that contains with fixed record fields.
  - Parse data in a file that contains delimited or fixed record fields using a regular expression.

  For an example of the use of the `DELIMITED` value, see the `SampleUniversal.xml` job file.

- **EXCHANGE2007**

  This value is used to parse data in a Microsoft Exchange Server 2007 message log file. For an example of the use of this value, see the `SampleExchange2007Log.xml` job file.

- **HMC**

  This value is used to collect data from HMC Server. For an example of the use of this value, see the `SampleHMC.xml` job file.

- **HMCINPUT**

  This value is used to parse data from HMC Server. For an example of the use of this value, see the `SampleHMC.xml` job file.

- **NFC**

  This value is used to parse data from NFC. For an example of the use of this value, see the `SampleNetworkFlow.xml` job file.

- **REGEX**

  This value is used to parse data in a record regular expression which is used to parse the record

- **TDS**

  This value is used to collect data from a TDSz DB2 database. For an example of the use of this value, see the `SampleTDSz.xml` job file.

- **TPC**

  This value is used to parse data in a TPC log file. For an example of the use of this value, see the `SampleTPC.xml` job file.

- **TPCVIEW**

  This value is used to collect data from TPC DB Data View. For an example of the use of this value, see the `SampleTPC_DiskLevel.xml` and `SampleTPC_StorageSubsystemLevel.xml` job files.

- **TRANSACTION**

  This value is used to collect data from the Transaction table in the SmartCloud Cost Management database. For an example of the use of this value, see the `SampleTransaction.xml` job file.

- **TSM**

  This value is used to collect data from a TSM DB2 database. For an example of the use of this value, see the `SampleTSM.xml` job file.

- **VMWARE**

  This value is used to collect data from VMware VirtualCenter or VMware Server Web service. For an example of the use of this value, see the `SampleVMWare.xml` job file.

- **WEBSPHEREXDFINEGRAIN**

This value is used to parse data in a WebSphere Extended Deployment (XD) Fine-Grained Power Consumption log file `FineGrainedPowerConsumptionStatsCache.log`. For an example of the use of this value, see the `SampleWebSphereXDFineGrain.xml` job file.

- **WEBSPHEREXDSERVER**

  This value is used to parse data in a WebSphere Extended Deployment (XD) Server Power Consumption log file `ServerPowerConsumptionStatsCache.log`. For an example of the use of this value, see the `SampleWebSphereXDFineGrain.xml` job file.

- The `File 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

This section describes the Stage elements.

Aggregator:

The `Aggregator` stage aggregates a file based on the identifiers and resources specified. Any resources and identifiers not specified are dropped from the record. The Aggregator stage is memory dependent; the amount of memory affects the amount of time it takes to perform aggregation.

Settings

The Aggregator stage accepts the following input elements.

Attributes

The following attributes can be set in the `<Aggregator>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of a resource to aggregate. This setting can be used multiple times to name multiple resources. If a resource is not named it will not be aggregated and will not be included in the output file.
- `aggregationfunction="sum | min | max | avg | none"`: The aggregation type. The default aggregation operator for resources is to sum all resource values. Additional aggregation functions are also available. The available aggregation functions are:
  - `SUM`: (the default) sums all resources producing a total
  - `MIN`: finds the minimum value of a resource within an aggregate
  - `MAX`: finds the maximum value of a resource within an aggregate
  - `AVG`: produces the average of a series of resource values within an aggregate
  - `NONE`: does not apply any aggregation function (useful for values such as the number of CPUs)

  This setting is used as follows:

```
<Resource name="RESSUM"  aggregationfunction = "avg"/>
```

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: This setting can be used multiple times to name multiple identifiers. If an identifier is not named it will not be aggregated and will not be included in the output file. The order in which records are placed in the output file is set by the order in which the identifiers are defined. Precedence is established in sequential order from the first identifier defined to the last. If a defined identifier appears in a record with a blank value, it will be included in the aggregated record with a blank value.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `defaultAggregation="true | false"`: This setting specifies whether the fields in the record header (start date, end date, account code, etc.) are used for aggregation. The default setting is true.
- `impliedZeroForMissingResources="true | false"`: There may be some scenarios that require a missing resource to be accounted for. This setting that allows you to add this behavior. For more information see "Implied zero for missing resources" on page 103.
- `includeNumRcdsResource="true | false"`: Setting this to true will cause the number of records within an aggregate to be counted using the includeNumRcdsResource option. A resource called Num_Rcds is generated listing the number. The default setting is false.
- `aggregationIntervalHours="num_hours"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of hours in that interval.
- `aggregationIntervalMinutes="num_minutes"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of minutes in that interval.
- `aggregationIntervalSeconds="num_seconds"`: It is possible to aggregate resource values into time intervals. This setting allows you to set the number of seconds in that interval.
- `aggregationIntervalValidation="true | false"`: Setting this to true will catch end time - start time values which exceed the specified aggregation interval. For example, if the aggregation interval is 5 minutes, and the record has data from a 10 minute interval, a warning is logged in the trace file.

Example

This is a basic example using the default aggregation operator, SUM. Assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `Aggregator` stage appears as follows:

```
<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:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXEMRCV,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXEMRCV,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,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. In the example, the identifier `User` is defined first. As a result, the output records are ordered based on user.

Aggregation by time interval:

In addition to aggregation functions, the aggregator can aggregate resource values into time intervals.

In addition to aggregation functions, the aggregator can aggregate resource values into time intervals. By default, time values in are ignored and only the identifiers specified in the `Identifiers` element are used to aggregate a series of records. However, if an aggregation interval value is specified, the aggregates will be grouped based on a time interval starting at midnight, based on the end time value. For example, if an aggregation interval of 5 minutes is specified, aggregator will create aggregates for resource records falling within the following time periods:

00:00:00 through 00:04:59

00:05:00 through 00:09:59

00:10:00 through 00:14:59

:

23:55:00 through 23:59:59

The end-time value of the incoming CSR record is used to determine which aggregate time slot the CSR record resource values are aggregated into it. The `aggregationIntervalValidation` parameter can be used to catch end-time to start-time values which exceed the specified aggregation interval. For example, if the aggregation interval is five minutes, and the record has data from a 10 minute interval, a warning is logged in the trace file.

The aggregation interval can be specified in seconds, minutes, or hours using the following parameters:

- `aggregationIntervalSeconds`

- aggregationIntervalMinutes
- aggregationIntervalHours

Implied zero for missing resources:

There may be some scenarios that require a missing resource to be accounted for. The setting that allows you to add this behavior is impliedZeroForMissingResources.

In some scenarios, one or more resources may not appear in all records. There is a default assumption that a resource missing from an input record means the resource is to be ignored for that record. However, there may be some scenarios that require that the missing resource to be accounted for. The option to control this behavior is impliedZeroForMissingResources, and is defined as:

```
<Parameter impliedZeroForMissingResources="false"/>
```

where "false", the default setting, assumes the metric is to be ignored, and "true" assumes the metric is to be accounted for.

If the option is set to true and a resource is missing from an input record, the resource is added with a value of zero.

This option primarily affects the calculation of the AVG function. For example, assume the following two input records having the resources number of CPUs (NUM_CPU) and memory used, (MEM_USED) and that the AVG function will be applied to both metrics:

```
Record #1: User1, NUM_CPU=2,  MEM_USED=4
Record #2: User1, MEM_USED=4
```

Notice in Record #2 the NUM_CPU resource value is missing. If impliedZeroForMissingResources is false (the default), the value of AVG (NUM_CPU) will be value 2 / 1 record = 2. If impliedZeroForMissingResources is true, the value of AVG (NUM_CPU) will be value 2 / 2 records = 1.

In effect, impliedZeroForMissingResources = "true" has the same effect as if the records were given as:

```
Record #1: User1, NUM_CPU=2,  MEM_USED=4
Record #2: User1, NUM_CPU=0, MEM_USED=4
```

CreateAccountRelationship:

The CreateAccountRelationship stage adds a client that represents an account in SmartCloud Cost Management. The stage also associates the newly created client with a rate table and a user group in SmartCloud Cost Management, creating the user group if it does not already exist. The stage updates existing client accounts, so they can be associated with other user groups and it can also modify the rate table that is used.

Settings

The CreateAccountRelationship stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateAccountRelationship> element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. The default setting is true.
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- `stopOnStageFailure = "true | false"`: Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `createGroup="true | false"`: Setting this to true means that the user group will be created if it does not exist. Setting this to false means that the user group will not be created if it does not exist. The default setting is true.

Mandatory Identifiers

The following attributes are expected in the CSR input data:

- `CLIENTNAME`: The client name that represents the account code. The client name is formed from the account code. If the client name already exists, then it can be updated for description, user group and rate table.
- `ACCOUNTCODE`: The account code that defines the structure, which reflects the chargeback hierarchy for the client.

Optional Identifiers

The following attributes are optional in the CSR input data:

- `USERGROUPID`: The id of the user group that the client is associated to. If the client account exists and it is not already associated, then the user groups associated with the client account are updated to this new group. A client account must be associated to at least one user group so that the client account is viable in SmartCloud Cost Management.
- `USERGROUPDESC`: Description of the user group that the client account is associated to. If `USERGROUPID` is specified and `USERGROUPDESC` is not, then the `USERGROUPID` value is used by default.
- `RATETABLE`: The rate table that the client account uses. If it is not specified, then the default `STANDARD` rate table is used. If the client account already exists, then the existing client rate table that is used is updated to this new table.
- `DEFAULTACS`: The default account code structure that the user group is associated with. If it is not specified, then the default `Standard` account code structure is used.

Example 1: New client account, user group does not exist

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,21,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_AT,2014-03-07 11:11:08,VM_LAUNCHED_A
ACCOUNTCODE,Admins ApplicationA                ,
CLIENTNAME,ApplicationA,USERGROUPID,ApplicationA Admins,USERGROUPDESC,Admin group for ApplicationA,5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DI
```

If the `CreateAccountRelationship` stage appears as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
      <Parameter createGroup="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- The client account "ApplicationA" is created.
- User group "ApplicationA Admins" is created and client account "ApplicationA" is associated with it.
- Client account uses "STANDARD" rate table.

Example 2: Existing client account, user group exists and rate table specified

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,22,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_AT,2014-03-07 11:11:08,VM_LAUNCHED_AT,20
ACCOUNTCODE,Admins ApplicationA                ,
CLIENTNAME,ApplicationA,USERGROUPID,ApplicationA User,USERGROUPDESC,User group for ApplicationA,RATETABLE,CUSTOM, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUM
```

If the CreateAccountRelationship stage appears as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
      <Parameter createGroup="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- Client account "ApplicationA" association is updated with the user group ApplicationA User.
- Client account "ApplicationA" is updated to use the CUSTOM rate table.

Considerations when using the CreateAccountRelationship stage

Consider the following when using the CreateAccountRelationship stage:

- The exception file can contain records for input data, where the user group does not exist and createGroup is set to false.
- The exception file can contain records for input data, where the STANDARD rate table does not exist in SmartCloud Cost Management when creating a user group with the default rate table.
- The exception file can contain records for input data, where the STANDARD account code structure does not exist in SmartCloud Cost Management when creating a user group with the default account code structure.
- The exceptionProcess parameter must be turned on for records that are written to the exception file. For more information, see the related topic in the Configuration guide.
- If CLIENTNAME is not specified in the CSR record, then the CSR record is added to the exception file.
- If ACCOUNTCODE is not specified in the CSR record, then the CSR record is added to the exception file.

CreateIdentifierFromIdentifiers:

The `CreateIdentifierFromIdentifiers` stage creates a new identifier by compounding the strings that comprise others. This can be done by taking whole identifier strings or by taking substrings.

Settings

The CreateIdentifierFromIdentifiers stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateIdentifierFromIdentifiers>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the new composite identifier created from existing identifiers.

FromIdentifiers:

The following attributes can be set in the `<FromIdentifier>` element:

- `name = "identifier_name"`: The name of an identifier that will be used to form part of the new identifier. The order of the `FromIdentifier` elements defines the order of concatenated values that appear in the new identifier value.
- `offset="numeric_value"`: This value, in conjunction with `length` allows you to set how much of the original identifier is used in creating the new identifier. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- `length="numeric_value"`: This value, in conjunction with `offset` allows you to set how much of the original identifier is used in creating the new identifier. If you want to use five characters of an identifer, the length is set to 5. This is set to 50 by default.
- `delimiter="delimiter_value"`: If set this value will be concatenated to the end of the FromIdentifier. This is null by default.

The following is an exmaple of a `FromIdentifier` setting:

```
<FromIdentifier name="User" offset="1" length="5" delimiter="a"/>
```

Parameters

The following attributes can be set in the `<Parameter>` element:

- keepLength="true | false" : The parameter keepLength specifies whether the entire length should be included. If the length specified is longer than the identifier value, the value is padded with spaces to meet the maximum length. The default for this setting is "false".
- modifyIfExists=="true | false": If this parameter is set to true, and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

A CreateIdentifierFromIdentifiers stage can be created as follows:

```
<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>
```

Running the preceding CreateIdentifierFromIdentifiers stage creates the following output CSR file.

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User, /
"joe",Account_Code,"joeaSrvr1b",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User, /
"mary",Account_Code,"maryaSrvr2b",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,/
"joan",Account_Code,"joanaSrvr3b",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,/
"joan",Account_Code,"joanaSrvr3b",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,/
"joe",Account_Code,"joeaSrvr1b",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.

CreateIdentifierFromRegEx:

The `CreateIdentifierFromRegEx` stage creates a new identifier the value of which is derived using a regular expression. The regular expression will derive the new value using an existing value.

Settings

The CreateIdentifierFromRegEx stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateIdentifierFromRegEx>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the new composite identifier created from existing identifiers.

FromIdentifier:

The following attributes can be set in the `<FromIdentifier>` element:

- `name ="identifier_name"`: The name of an identifier that will be used to form part of the new identifier.
- `regEx="regular_expression"`: The regular expression used to parse the identifier value.
- `value="regular_expression_group"`: The value of the segment that will be used to create the new identifier. Please see the example for more information.

Parameters

The following attributes can be set in the `<Parameter>` element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joe.allen@xyz.com",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"mary.kay@xyz.com",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joan.jet@xyz.com",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joan.jet@xyz.com",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,1,EmailID,"joe.allen@xyz.com",2,EXEMRCV,1,EXBYRCV,2817
```

If the CreateIdentifierFromRegEx stage appears as follows:

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

The output CSR file appears as follows:

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joe.allen@xyz.com", \
FirstName,"joe",LastName,"allen",FullName,"allen, joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"mary.kay@xyz.com", \
FirstName,"mary",LastName,"kay",FullName,"kay, mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joan.jet@xyz.com", \
FirstName,"joan",LastName,"jet",FullName,"jet, joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joan.jet@xyz.com", \
FirstName,"joan",LastName,"jet",FullName,"jet, joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,4,EmailID,"joe.allen@xyz.com", \
FirstName,"joe",LastName,"allen",FullName,"allen, joe",2,EXEMRCV,1,EXBYRCV,2817
```

The identifiers FirstName, LastName, and FullName were added.

CreateIdentifierFromTable:

The CreateIdentifierFromTable stage creates a new identifier from the values defined in the conversion table.

Settings

The CreateIdentifierFromTable stage accepts the following input elements.

Attributes:

The following attributes can be set in the <CreateIdentifierFromTable> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)

- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": The name of the new composite identifier created from another identifier's value as a lookup to a conversion table. Only one new identifier can be specified in the stage.

FromIdentifier:

The following attributes can be set in the <FromIdentifier> element:

- name = "identifier_name": The name of an identifier the value of which will be sought in the conversion table. If a match is not found in the conversion process, the Identifier will be added with a value of spaces unless the exceptionProcess is turned on. In that case, the record will be written to the exception file.
- offset="numeric_value": This value, in conjunction with length allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- length="numeric_value": This value, in conjunction with offset allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifer, the length is set to 5. This is set to 50 by default.

Files:

The following attributes can be set in the <File> element:

- name = "file_name": This is set to the name of the file that contains the conversion table. The number of definition entries that you can enter in the conversion table is limited only by the memory available to Integrator.
- type="table | exception": There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- encoding="system | encoding_scheme": This option is available only if the type is set to table. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, e.g., UTF-8.
- format="CSROutput | CSRPlusOutput": This option is available only if the type is set to exception. The exception file can be in any output format that is supported by 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.

DBLookups:

The <DBLookup> element over rides the default functionality of loading the conversion table from file and loads the conversion mappings from the SmartCloud Cost Management database instead. The following attributes can be set in the <DBLookup> element:

- process = "process_name": The name of the Process Definition corresponding to the conversion mappings that you want to reference. If this is not set, the Process Id of the job file is used as the default.
- discrimator="first | last | largest": When the discrimator attribute is set to "first", it references the first conversion entry that matches the usage period. When set to "last", it references the last conversion entry that matches the usage period. When set to "largest", it references the conversion entry which matches the largest timeline for the usage period. If this is not set, the default is last.
- cacheSize="integrer value between 1 and 99": This configures the maximum number of periods that may be cached in memory for processing the CSR input. Each distinct usage period in the CSR input necessitates a lookup in the database. These periods are cached for subsequent records. Setting the cacheSize to a high value will improve performance of the stage but will use more memory. Set to 24 by default.

Parameters:

The following attributes can be set in the <Parameter> element:

- exceptionProcess="true | false" : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry.
- sort="true | false" : If the parameter sort is set to true (the default and recommended), an internal sort of the conversion table is performed.
- upperCase="true | false" : The conversion table is case-sensitive. For convenience, you can enter uppercase values in the conversion table for your identifiers and then set the parameter upperCase="true". This ensures that identifier values in your CSR input data that are lowercase or mixed case are processed. The default for upperCase is false.
- writeNoMatch="true | false" : If this 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 setting is false.
- modifyIfExists=="true | false": If this parameter is set to true and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example - File

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",
Account_Code,"ATM",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",
Account_Code,"CCX",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,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`.

Example - DBLookup

Example 1: discriminator = "first"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the `CreateIdentifierFromTable` stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="first"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 1:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"John",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Example 2: discriminator = "last"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the CreateIdentifierFromTable stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="last"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 2:

```
VVMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Pat",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Example 3: discriminator = "largest"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,2,HostName,"esx1.example.com",VMName,"VM1",2,
VMCPUUSE,98301,VMCPUGUA,239
```

If the CreateIdentifierFromTable stage appears as follows:

```
<Stage name="CreateIdentifierFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="largest"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59','Pat'
```

The output CSR file is displayed as follows for example 3:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",Account_Code,
"Paul",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Note: Refer to the sample jobfile SampleAutomatedConversions.xml when using this stage.

CreateIdentifierFromValue:

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

Settings

The CreateIdentifierFromValue stage accepts the following input elements.

Attributes:

The following attributes can be set in the <CreateIdentifierFromValue> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": The name of the new identifier.
- value=" value": The value that is attributed to the new identifier.

Parameters :

The following attributes can be set in the <Parameter> element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the CreateIdentifierFromValue stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr2",User,"mary",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr3",User,"joan",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,3,Feed,"Srvr1",User,"joe",Break_Room,"North",2,EXEMRCV,1,EXBYRCV,2817
```

The identifier Break_Room was added with a value of North.

CreateResourceFromConversion:

The CreateResourceFromConversion stage creates a new resource, with the value derived using an arithmetic expression.

Settings

The CreateResourceFromConversion stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateResourceFromConversion> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)

- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not exist in the table.
- `symbol="a-z"`: This allows you to pick a variable which can be used to represent the value of the new resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

FromResource

The following attributes can be set in the `<FromResource>` element:

- `name = "resource_name"`: The name of an existing resource with the value used to derive a new resource value.
- `symbol="a-z"`: This allows you to pick a variable which will be given the value of the named resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

Parameters

The following attributes can be set in the `<Parameter>` element:

- `formula = "arithmetic_expression"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element. In addition minimum and maximum capability is provided. Rounding functions are also provided based on the Java Rounding Modes definition.
- `modifyIfExists=="true | false"`: If this parameter is set to true and the identifier exists, the existing identifier value is modified with the specified value. If this is set to false (the default), the existing identifier value is not changed.

Example 1

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Total_Resource">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   <FromResource name="EXBYRCV" symbol="b"/>
   </FromResources>
   </Resource>
   </Resources>
   <Parameters>
```

```
   <Parameter formula="(a+b)/60"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Total_Resource,65.7
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Total_Resource,64.4
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Total_Resource,45.81667
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Total_Resource,50.23333
Example,20070117,20070117,00:00:00,23:59:59,1,2,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.

Example 2

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,3092,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,3000,EXBYRCV,2817
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Max_Resource">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   <FromResource name="EXBYRCV" symbol="b"/>
   </FromResources>
   </Resource>
   </Resources>
   <Parameters>
   <Parameter formula="max(a,b)"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Max_Resource,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Max_Resource,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Max_Resource,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,3092,EXBYRCV,3013,Max_Resource, 3092
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,3000,EXBYRCV,2817,Max_Resource, 3000
```

The resource `Max_Resource` was added. The value for the new resource is built from the maximum of the existing resource values.

Example 3

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",1,EXEMRCV,1.6
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",1,EXEMRCV,1.5
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"john",1,EXEMRCV,1.2
```

If the `CreateResourceFromConversion` stage appears as follows:

```
<Stage name="CreateResourceFromConversion" active="true">
   <Resources>
   <Resource name="Half_Up_Res">
   <FromResources>
   <FromResource name="EXEMRCV" symbol="a"/>
   </FromResources>
   </Resource>
   </Resources>
```

```
   <Parameters>
   <Parameter formula="HALF_UP(a)"/>
   <Parameter modifyIfExists="true"/>
   </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1.6,Half_Up_Res,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1.5,Half_Up_Res,2
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"john",2,EXEMRCV,1.2,Half_Up_Res,1
```

The resource `Half_Up_Res` was added. The value for the new resource is built from the `HALF_UP` of the existing resource values.

Considerations for Using CreateResourceFromConversion

Consider the following when using the `CreateResourceFromConversion` stage:

- Rounding functionality supported based on the Java Rounding Modes definition as follows:
  - **`CEILING:`** rounding mode to round towards positive infinity.
  - **`DOWN:`** rounding mode to round towards zero.
  - **`FLOOR:`** rounding mode to round towards negative infinity.
  - **`HALF_DOWN:`** rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.
  - **`HALF_EVEN:`** rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor.
  - **`HALF_UP:`** rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.
  - **`UNNECESSARY:`** rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary.
  - **`UP:`** rounding mode to round away from zero.
- Minimum and maximum capability is on two Resource elements only and does not include arithmetic expressions in it.
- Rounding capability is on one Resource element only and does not include arithmetic expressions in it.

CreateResourceFromDuration:

The `CreateResourceFromDuration` stage creates a new resource the value of which is set by calculating the difference between the start time and the end time in a CSR or CSR+ record. These times can be taken from the time fields in the record (start date, end date, start time, end time) or they can be taken from the identifier values in the record. Once the new duration resource has been created, it can be used in subsequent Integrator stages by using a mathematical formula to modify other resources.

Settings

The `CreateResourceFromDuration` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateResourceFromDuration>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: The name of the new resource. If the Duration calculation results in a value of 0 for the units specified, then no Resource is created. You must add the duration resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.

FromDateTimes:

The following attributes can be set in the `<FromDateTime>` element:

- `name = "identifier_name"`: The name of the new resource.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `units = "milliseconds | seconds | minutes | hours | days"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element. Duration calculations use truncation, so partial units get dropped. For example, if the CSR record shows a duration of 2.5 hours, and the unit specified is hours, the Duration resource will be written as “2”.
- `dataFormat=java_date_format` : This paramter provides the format which will be used to interpret the timestamp values. The parameter dateFormat uses the conventions described by Java's SimpleDateFormat class. See its javadoc for examples of how to specify a date format.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example 1: Creating a resource using the time fields

In this example, assume that the following CSR file is the input and that the output file is also defined as a CSR file.

```
Example,20080117,20080117,08:00:00,08:00:45,,1,Feed,Srvr1,2,EXEMRCV,1,EXBYRCV,3941
Example,20080117,20080117,09:00:00,09:30:00,,1,Feed,Srvr2,2,EXEMRCV,1,EXBYRCV,3863
Example,20080117,20080117,10:00:00,14:30:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,2748
Example,20080117,20080217,08:00:00,06:00:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,3013
```

If the `CreateResourceFromDuration` stage appears as follows:

```
<Stage name="CreateResourceFromDuration" active="true">
       <Resources>
          <Resource name="Duration">
          </Resource>
       </Resources>
       <Parameters>
          <Parameter units="seconds"/>
          <Parameter modifyIfExists="true"/>
       </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20080117,20080117,08:00:00,08:00:45,,1,Feed,Srvr1,2,EXEMRCV,1,EXBYRCV,3941,Duration,45
Example,20080117,20080117,09:00:00,09:30:00,,1,Feed,Srvr2,2,EXEMRCV,1,EXBYRCV,3863,Duration,1800
Example,20080117,20080117,10:00:00,14:30:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,2748,Duration,9000
Example,20080117,20080217,08:00:00,06:00:00,,1,Feed,Srvr3,2,EXEMRCV,1,EXBYRCV,3013,Duration,79200
```

The start time and end time are calculated using the time fields in the CSR or CSR+ record. For example, in the first record, the start and end duration is 45 seconds. As the `units` parameter value is `"seconds"`, a resource named `Duration` was created with a value of `45`. In the second record, the start and end duration is 30 minutes. Therefore, the `Duration` resource value is `1800` seconds.

Example 2: Creating a resource using identifier values

In this example, assume that the following CSR file is the input and that the output file is also defined as a CSR file.

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20080117,20080117,08:00:00,08:00:45,,3,Feed,Srvr1,StartDate,"2007/08/19  16:00:00", \
EndDate," 2007/08/19  16:30:15",01,LLY202,1846
Example,20080117,20080117,09:00:00,09:30:00,,3,Feed,Srvr2,StartDate,"2007/08/19  19:00:00", \
EndDate," 2007/08/19  19:00:15",01,LLY202,1846
```

If the `CreateResourceFromDuration` stage appears as follows:

```
<Stage name="CreateResourceFromDuration" active="true">
       <Resources>
          <Resource name="UserSpecifiedDuration">
             <FromDateTimes>
                <FromDateTime name="StartDate"/>
                <FromDateTime name="EndDate"/>
             </FromDateTimes>
          </Resource>
       </Resources>
       <Parameters>
          <Parameter dateFormat="yyyy/MM/dd HH:mm:ss"/>
          <Parameter units="minutes"/>
          <Parameter modifyIfExists="true"/>
       </Parameters>
    </Stage>
```

The output CSR file appears as follows:

Note: Due to the length of the CSR lines, each line has been split and a backslash has been placed at the point of the split.

```
Example,20080117,20080117,08:00:00,08:00:45,1,3,Feed,"Srvr1",StartDate,"2007/08/19 16:00:00",EndDate, \
" 2007/08/19 16:30:15", 2,LLY202,1846,UserSpecifiedDuration,30
Example,20080117,20080117,09:00:00,09:30:00,,3,Feed,Srvr2,StartDate,"2007/08/19  19:00:00",EndDate, \
" 2007/08/19  19:00:15",01,LLY202,1846
```

In this example, the `FromDateTime` elements specify the identifier names to use for calculating the duration (`StartDate` and `EndDate`). The `dateFormat` parameter provides the format that will be used to interpret the timestamp values. In the first record, the duration is 30 minutes and 15 seconds and the `units` parameter is `"minutes"`, so a resource named `UserSpecifiedDuration` was created with a value of `30`. Duration calculations use truncation, so the partial units of 15 seconds are dropped.

In the second record, the duration is 15 seconds, and the units are specified as minutes. Because the units value is in minutes and values are truncated, the calculation results in a value of 0 and a `UserSpecifiedDuration` resource was not created in this record.

CreateResourceFromValue:

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

Settings

The `CreateResourceFromValue` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<CreateResourceFromValue>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "resource_name"`: The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.
- `value=" numeric_value"`: The new resource value.

Parameters :

The following attributes can be set in the `<Parameter>` element:

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,3941,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",3,EXEMRCV,1,EXBYRCV,3863,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,2748,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
```

The resource Num_Recs was added with a value of 1.

CreateUserRelationship:

A user is an individual with access rights to SmartCloud Cost Management web-based applications. Each user can belong to one or more user groups. Users are granted the rights and privileges that are granted to the group. The CreateUserRelationship stage adds a user and associates that user to a user group in SmartCloud Cost Management. The stage can also update existing users to be associated to other user groups.

Settings

The CreateUserRelationship stage accepts the following input elements.

Attributes

The following attributes can be set in the <CreateUserRelationship> element:

- active = "true | false": Setting this attribute to true activates this stage within a job file. The default setting is true.
- trace = "true | false" : Setting this attribute to true enables the output of trace lines for this stage. The default setting is false.
- stopOnStageFailure = "true | false": Setting this attribute to true halts the execution of this stage if an error occurs. The default setting is true.

Files

The following attributes can be set in the <File> element:

- name = "file_name": This is set to the name of the exception file.
- type = "exception": The type and then the format or encoding must be specified.
- format = "CSROutput | CSRPlusOutput": This option is only available if the type is set to exception. The exception file can be in any output format that is supported by Integrator. The format is defined by the stage name of the output type. For example, if the stage CSROutput or CSRPlusOutput is active, the exception file is produced as CSR or CSR+ file.

Mandatory Identifiers

The following attributes are expected in the CSR input data:

- USERID: The identifier of the user.

Optional Identifiers

The following identifiers are optional in the CSR input data:

- USERDESC: Description of the user which is used as full name. If USERDESC is not specified, then the USERID value is used by default.
- USERGROUPID: The ID of the user group that the client is associated to. If the user exists and the user is not already associated, then the user groups associated with that user are updated to this new group. If the user already exists in the database, you may want to associate other groups to that user. If however, you

don't specify a group and the user is already in the database, then it will have a group already associated to it. In this scenario, the record can be ignored.

•

Parameters

- `defaultUserGroup="<userGroupId>"`: The Id of the user group which is used as a default in some scenarios when group is not defined in CSR file.

Example 1: New user, user group exists

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,19,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_A
USERID,john.smith, USERGROUPID,ApplicationA Admins, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DISK,0
```

If the `CreateUserRelationship` stage is displayed as follows:

```
<Stage name="CreateUserRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- The user `john.smith` is created.
- User is now associated to the `ApplicationA Admins` user group.

Example 2: Existing user, user group exists

Assume that the following CSR file is the input:

```
SCO,20150219,20150219,00:00:00,00:00:00,1,19,VM_ID,2073287d-de51-4e2b-b4e6-a82754a616cb,VM_NAME,VM091606391,VM_CREATED_A
USERID,john.smith, USERGROUPID,ApplicationB Admins, 5,USEDURN,1440,VMSTATIP,1,VM_MEMORY,512,VMNUMCPU,1,VM_DISK,0
```

If the `CreateUserRelationship` stage is displayed as follows:

```
<Stage name="CreateAccountRelationship" active="true">
   <Files>
      <File name="Exception.txt" type="exception" format="CSROutput"/>
   </Files>
   <Parameters>
      <Parameter exceptionProcess="true"/>
   </Parameters>
</Stage>
```

The result is as follows:

- User `john.smith` is now associated to the `ApplicationB Admins` user group.

Considerations when using the `CreateUserRelationship` stage

Consider the following when using the `CreateUserRelationship` stage:

- If you are creating clients and user groups with users from this `CreateUserRelationship`, the stage must be called after the `CreateAccountRelationship` stage.
- If the user is associated to an `admin` group in the stage, it is disassociated from all previous groups as a consequence.
- The `exceptionProcess` parameter must be turned on for records that are written to the exception file. For more information, see the *Enabling exception processing* topic in the Configuration guide.

- If no user group is specified and the user does not exist, then add a CSR record to the exception file.
- If no user group is specified and the user exists, then ignore the CSR record.
- If no user group is specified, the default user group is configured, and the user exists, then ignore the CSR record.
- If USERID is not specified in the CSR record, then the CSR record is added to the exception file.

CSROutput:

The CSROutput stage produces a CSR file.

Parameters:

The following attributes can be set in the <parameter> element:

- <Parameter keepZeroValueResources="true | false"/>: This parameter when set to true, enables resources with zero values to be written to CSR files and billing output or read from CSR files and billing output. Resources with zero values are normally discarded.

Example

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

In this example, the CSR 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.

Parameters:

The following attributes can be set in the <parameter> element:

- <Parameter keepZeroValueResources="true | false"/>: This parameter when set to true, enables resources with zero values to be written to or read from CSR files and in billing output. Resources with zero values are normally discarded.

Example

```
<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.

DropFields:

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

Settings

The DropFields stage accepts the following input elements.

Attributes:

The following attributes can be set in the <DropFields> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Fields:

The following attributes can be set in the <Field> element:

- name = "resource_name | identifier_name": This sets the name of the resource or identifier to drop.

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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the DropFields stage appears as follows:

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

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"mary",1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",1,EXBYRCV,2817
```

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

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.

Settings

The DropIdentifiers stage accepts the following input elements.

Attributes:

The following attributes can be set in the <DropIdentifiers> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": This sets the name of the identifier to drop.

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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817
```

If the DropIdentifiers stage appears as follows:

```
<Stage name="DropIdentifiers" active="true">
   <Identifiers>
   <Identifier name="Feed">
   </Identifiers>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joe",2,Feed,1,EXBYRCV,2817
```

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

DropResources:

The `DropResources` stage drops a specified resource from a 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.

Settings

The `DropResources` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<DropResources>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"`: This sets the name of the resource to drop.

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.

Note: 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.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,Feed,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,Feed,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,Feed,1,EXBYRCV,2817
```

If the `DropResources` stage appears as follows:

```
<Stage name="DropResources" active="true">
   <Resources>
   <Resource name="Feed"/>
   </Resources>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",1,EXBYRCV,2817
```

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

ExcludeRecsByDate:

The `ExcludeRecsByDate` stage excludes 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.

Settings

The `ExcludeRecsByDate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<ExcludeRecsByDate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the `<Parameter>` element:

- `fromDate = "from_date"`: This parameter allows you to specify the exclusion start date. The date is entered in the format YYYYMMDD.
- `toDate= "to_date"`: This parameter allows you to specify the exclusion end date. The date is entered in the format YYYYMMDD.
- `keyWord= "**PREDAY | **CURDAY | **RNDATE | **PREMON | **CURMON | **PREWEK | **CURWEK"`: This parameter allows you to use a date keyword to specify the exclusion date range.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the ExcludeRecsByDate stage appears as follows:

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

Or

```
<Stage name="ExcludeRecsByDate" active="true">
   <Parameters>
   <Parameter fromDate="20070101"/>
   <Parameter toDate="20070131"/>
   </Parameters>
</Stage>
```

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

```
Example,20070217,20070217,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

Only those records with end dates in February are included.

ExcludeRecsByPresence:

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

Settings

The ExcludeRecsByPresence stage accepts the following input elements.

Attributes:

The following attributes can be set in the <ExcludeRecsByPresence> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the <Resource> element:

- name = "resource_name" : This parameter allows you to specify the resource that will be tested for existence and excluded based on the following condition.
- exists="true | false": Setting this parameter to true will cause a record to be dropped if it contains the named resource. Setting it to false will cause a record to be dropped if it does not contain the resource.

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name" : This parameter allows you to specify the identifier that will be tested for existence and excluded based on the following condition.
- exists="true | false": Setting this parameter to true will cause a record to be dropped if it contains the named identifier. Setting it to false will cause a record to be dropped if it does not contain the identifier.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple entries are treated as OR conditions. If any one of the conditions is met, the record is dropped.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",2,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,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,1,1,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.

ExcludeRecsByValue:

The ExcludeRecsByValue stage drops records based on an identifier values, resource values, or both.

Settings

The ExcludeRecsByValue stage accepts the following input elements.

Attributes:

The following attributes can be set in the <ExcludeRecsByValue> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)

- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"` : This setting allows you to enter the name of a resource for comparison. If the following comparison conditions are met, the record containing this identifier will not be included in the output.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. 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)
  - LIKE (starts with, ends with, and contains a string. Starts with the value format = `"string%"`, ends with the value format = `"%string"`, and contains the value format = `"%string%"`)
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is excluded from the output file.

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"` : This setting allows you to enter the name of an identifier for comparison. If the following comparison conditions are met, the record containing this identifier will not be included in the output.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. The comparison conditions have been stated previously.
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is excluded from the output file.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. 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 dropped.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `ExcludeRecsByValue` stage appears as follows:

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

The output CSR file appears as follows:

```
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,
EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,
EXBYRCV,3863
```

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

FormatDateIdentifier:

The `FormatDateIdentifier` stage allows you to reformat a date type identifier.

Settings

The `FormatDateIdentifier` stage accepts the following input elements:

Attributes:

The following attributes can be set in the `<FormatDateIdentifier>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Identifiers:

The following attribute can be set in the `<Identifier>` element:

- name = `"identifier_name"`: The name of the date identifier to be reformatted.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `inputFormat = "java date_time pattern"`: This parameter specifies the format of the input date identifier.
- `outputFormat = "java date_time pattern"`: This parameter specifies the format of the output date identifier.

Example

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

```
Example,20070602,,10:00:00,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI00001,Jobname,ALIREC6,
Start_date,20070602,Shift,1,20,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,1.31,Z005,1708,Z006,1367,
Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,1333,Z012,10270,Z013,25987,
Num_Rcds,5
```

```
Example,20070602,,11:47:56,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,BLI00002,Jobname,ABCDLYBK,
Start_date,20070602,Shift,1,19,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,Z005,93160,Z006,
4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,Z011,258557,Z012,494924,
Z013,7950768,Num_Rcds,1

Example,20070602,,11:40:25,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,CLI00003,Jobname,ABCOPER,
Start_date,20070602,Shift,1,17,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,13,Z006,13,
Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284,Num_Rcds,4
```

If the `FormatDateIdentifier` stage appears as follows:

```
<Stage name="FormatDateIdentifier" active="true" trace="false" >
 <Identifiers>
 <Identifier name="Start_date"/>
 </Identifiers>
 <Parameters>
 <Parameter inputFormat="yyyyMMdd"/>
 <Parameter outputFormat="dd.MM.yyyy"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
Example,20070602,20070602,10:00:00,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"ALI00001",
Jobname,"ALIREC6",Start_date,"02.06.2007",Shift,"1",19,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,
1.31,Z005,1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,
1333,Z012,10270,Z013,25987

Example,20070602,20070602,11:47:56,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"BLI00002",Jobname,
"ABCDLYBK",Start_date,"02.06.2007",Shift,"1",18,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,
Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,Z011,258557,
Z012,494924,Z013,7950768

Example,20070602,20070602,11:40:25,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"CLI00003",Jobname,
"ABCOPER",Start_date,"02.06.2007",Shift,"1",16,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,
13,Z006,13,Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284
```

The format of the identifier `Start_date` was changed to `dd.MM.yyyy` in the output CSR file.

IdentifierConversionFromTable:

The `IdentifierConversionFromTable` stage converts an identifiers value using the identifiers own value or another identifiers value as a lookup to a conversion table.

Settings

The `IdentifierConversionFromTable` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<IdentifierConversionFromTable>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The name of the identifier created by converting the value of the identifier using the identifiers own value or another identifiers value as a lookup to a conversion table. If the identifier defined for conversion is not found in the input record, the record is treated as an exception record. Only one new identifier can be specified per stage.

FromIdentifier

The following attributes can be set in the <FromIdentifier> element:

- name = "identifier_name": The name of an identifier the value of which will be sought in the conversion table. If a match is not found in the conversion process, the Identifier will be added with a value of spaces unless the exceptionProcess is turned on. In that case, the record will be written to the exception file.
- offset="numeric_value": This value, in conjunction with length allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you were going to use the whole identifier string, your offset would be set to 1. However, if you were selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- length="numeric_value": This value, in conjunction with offset allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifier, the length is set to 5. This is set to 50 by default.

Files

The following attributes can be set in the <File> element:

- name = "file_name": This is set to the name of the file that contains the conversion table. The number of definition entries that you can enter in the conversion table is limited only by the memory available to Integrator.
- type="table | exception": There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- encoding="system | encoding_scheme": This option is available only if the type is set to table. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, for example, UTF-8.
- format="CSROutput | CSRPlusOutput": This option is available only if the type is set to exception. The exception file can be in any output format that is supported by 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.

DBLookups

The <DBLookup> element over rides the default functionality of loading the conversion table from file and loads the conversion mappings from the SmartCloud Cost Management database instead. The following attributes can be set in the <DBLookup> element:

- process = "process_name": The name of the Process Definition corresponding to the conversion mappings that you want to reference. If this is not set, the Process Id of the job file is used as the default.
- discrimator="first | last | largest": When the discrimator attribute is set to "first", it references the first conversion entry that matches the usage period. When set to "last", it references the last conversion entry that matches the usage period. When set to "largest", it references the conversion entry which matches the largest timeline for the usage period. If this is not set, the default is last.
- cacheSize="integrer value between 1 and 99": This configures the maximum number of periods that may be cached in memory for processing the CSR input.

Each distinct usage period in the CSR input necessitates a lookup in the database. These periods are cached for subsequent records. Setting the `cacheSize` to a high value will improve performance of the stage but will use more memory. Set to `24` by default.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `exceptionProcess="true | false"` : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry. Records will not be written to the exception file.
- `sort="true | false"` : If the parameter sort is set to "true" (the default and recommended), an internal sort of the conversion table is performed.
- `upperCase="true | false"` : 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".
- `writeNoMatch="true | false"` : If this 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 setting is false.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example - File

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoe",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerMary",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"ServerJoan",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,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`.

Example - DBLookup

Example 1: discriminator = "first"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the `IdentifierConversionFromTable` stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="first"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 1:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"John",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

Example 2: discriminator = "last"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,
"VM1",Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the IdentifierConversionFromTable stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="last"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 2:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Pat",2,VMCPUUSE,98301,VMCPUGUA,239
```

The value for the VMName identifier was used to determine the new value for the Account_Code identifier as defined by the effective conversions defined in the VMWARE process.

Example 3: discriminator = "largest"

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

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,
"VM1",Account_Code,"James",2,VMCPUUSE,98301,VMCPUGUA,239
```

If the IdentifierConversionFromTable stage appears as follows:

```
<Stage name="IdentifierConversionFromTable" active="true">
 <Identifiers>
 <Identifier name="Account_Code">
 <FromIdentifiers>
 <FromIdentifier name="VMName" offset="1" length="7"/>
 </FromIdentifiers>
 </Identifier>
 </Identifiers>
 <Files>
 <File name="Exception.txt" type="exception" format="CSROutput"/>
 </Files>
 <DBLookups>
 <DBLookup process="VMWARE" discriminator="largest"/>
 </DBLookups>
 <Parameters>
 <Parameter exceptionProcess="true"/>
 <Parameter sort="false"/>
 <Parameter upperCase="false"/>
```

```
 <Parameter writeNoMatch="false"/>
 <Parameter modifyIfExists="false"/>
 </Parameters>
</Stage>
```

And the conversion mappings are defined as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-25 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-26 00:00:00','2199-12-31 23:59:59',Pat'
```

The output CSR file is displayed as follows for example 3:

```
VMWARE,20120101,20120131,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",
Account_Code,"Paul",2, VMCPUUSE,98301,VMCPUGUA,239
```

The value for the `VMName` identifier was used to determine the new value for the `Account_Code` identifier as defined by the effective conversions defined in the `VMWARE` process.

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.
- 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 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.
- 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"`.
- 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 exists, the existing identifier value is modified with the specified value. If `modifyIfExists="false"` (the default) existing identifier value is not changed.
- If the parameter `discrimator` is set to..
  - `last`(the default), it references the last conversion entry that matches the usage period.
  - `first`, it references the first conversion entry that matches the usage period.
  - `largest`, it references the conversion entry which matches the largest timeline for the usage period.

- 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 (fore example, ",,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 Integrator.

Note: Refer to the sample jobfile `SampleAutomatedConversions.xml` when using this stage.

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.

Settings

The `IncludeRecsByDate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByDate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the `<Parameter>` element:

- `fromDate = "from_date"`: This parameter allows you to specify the inclusion start date. The date is entered in the format YYYYMMDD.
- `toDate= "to_date"`: This parameter allows you to specify the inclusion end date. The date is entered in the format YYYYMMDD.
- `keyWord= "**PREDAY | **CURDAY | **RNDATE | **PREMON | **CURMON | **PREWEK | **CURWEK"`: This parameter allows you to use a date keyword to specify the inclusion date range.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070217,20070217,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `IncludeRecsByDate` stage appears as follows:

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

Or

```
<Stage name="IncludeRecsByDate" active="true">
   <Parameters>
   <Parameter fromDate="20070101"/>
   <Parameter toDate="20070131"/>
   </Parameters>
</Stage>
```

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

```
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
```

Only those records with end dates in January are included.

IncludeRecsByPresence:

The `IncludeRecsByPresence` stage includes records based on the existence or non-existence of identifiers, resources, or both. The resource or identifier name can also be defined using regular expressions, so you can use wildcards and not need an exact match for the identifier or resource name.

Settings

The `IncludeRecsByPresence` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByPresence>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name_regular_expression"` : This parameter allows you to specify either the exact resource name or a regular expression for the resource that will be tested for existence and included based on the following condition.
- `exists = "true | false"`: Setting this parameter to true will cause a record to be included if it contains the named resource. Setting it to false will cause a record to be dropped if it contains the named resource.

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name_regular_expression" : This parameter allows you to specify either the exact identifier name or a regular expression for the identifier that will be tested for existence and included based on the following condition.
- exists = "true | false": Setting this parameter to true will cause a record to be included if it contains the named identifier. Setting it to false will cause a record to be dropped if it contains the named identifier.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple entries are treated as OR conditions. If any one of the conditions is met, the record is included.

Example 1

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",3,EXEMRCV,1,EXBYRCV,3013,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joe",3,EXEMRCV,1,EXBYRCV,2817,Num_Recs,1
Example,20070117,20070117,00:00:00,23:59:59,,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,1,1,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,1,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.

Example 2

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

```
Example,20070602,,10:00:00,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,ALI00001,Jobname,
ALIREC6,Start_date,20070602,Shift,1,20,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,Z033,1.31,Z005,
1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,Z010,14876,Z011,1333,
Z012,10270,Z013,25987,Num_Rcds,5

Example,20070602,,11:47:56,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,BLI00002,Jobname,
```

```
ABCDLYBK,Start_date,20070602,Shift,1,19,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,62.3,Z033,66.6,
Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,Z010,753696,
Z011,258557,Z012,494924,Z013,7950768,Num_Rcds,1

Example,20070602,,11:40:25,,1,06,System_ID,ALIJ,Work_ID,JES2,Account_Code,CLI00003,Jobname,ABCOPER,
Start_date,20070602,Shift,1,17,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,Z033,0.3,Z005,13,Z006,13,
Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284,Num_Rcds,4
```

If the `IncludeRecsByPresence` stage appears as follows:

```
<Stage name="IncludeRecsByPresence" active="true" trace="true" >
   <Resources>
   <Resource name="[ZZ][0-9][0-9].*" exists="true">
   </Resources>
 </Stage>
```

The output CSR file appears as follows:

```
The output CSR file appears as follows:
Example,20070602,20070602,10:00:00,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"ALI00001",
Jobname,"ALIREC6",Start_date,"20070602",Shift,"1",19,Z001,1,Z002,4,Z003,1,Z031,0.91,Z032,1.27,
Z033,1.31,Z005,1708,Z006,1367,Z007,341,Z008,1367,Z011,341,Z014,13,ZZ05,1,ZZ06,5,Z009,52466,
Z010,14876,Z011,1333,Z012,10270,Z013,25987

Example,20070602,20070602,11:47:56,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"BLI00002",
Jobname,"ABCDLYBK",Start_date,"20070602",Shift,"1",18,Z001,1,Z002,1,Z003,62.13,Z031,46.25,Z032,
62.3,Z033,66.6,Z005,93160,Z006,4036,Z007,89124,Z008,4036,Z011,89124,ZZ05,6,ZZ06,19,Z009,9457945,
Z010,753696,Z011,258557,Z012,494924,Z013,7950768

Example,20070602,20070602,11:40:25,,1,6,System_ID,"ALIJ",Work_ID,"JES2",Account_Code,"CLI00003",
Jobname,"ABCOPER",Start_date,"20070602",Shift,"1",16,Z001,2,Z002,3,Z003,0.16,Z031,0.15,Z032,0.3,
Z033,0.3,Z005,13,Z006,13,Z008,13,Z014,28,ZZ06,3,Z009,6523,Z010,2416,Z011,213,Z012,610,Z013,3284
```

The three records in the input file were included because they contain a resource which matches the regular expression i.e. ZZ followed by 2 or more digits.

IncludeRecsByValue:

The `IncludeRecsByValue` stage includes records based on identifier values, resource values, or both. If the comparison is true, the record is included.

Settings

The `IncludeRecsByValue` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<IncludeRecsByValue>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the `<Resource>` element:

- `name = "resource_name"` : This setting allows you to enter the name of a resource for comparison. If the following comparison conditions are met, the record will be included in the output. If a field specified for inclusion contains a blank value, the record is included.

- `cond="GT | GE | EQ |LT | LE | LIKE"` : This setting allows you to enter the comparison condition. 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)
  - LIKE (starts with, ends with, and contains a string. Starts with the value format = `"string%"`, ends with the value format = `"%string"`, and contains the value format = `"%string%"`)
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is included in the output file.

Identifiers:

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"` : This setting allows you to enter the name of an identifier for comparison. If the following comparison conditions are met, the record will be included in the output. If a field specified for inclusion contains a blank value, the record is included.
- `cond="GT | GE | EQ |LT | LE | LIKE"`: This setting allows you to enter the comparison condition. The comparison conditions have been stated previously.
- `value="numeric_value"`: This setting allows enter the comparison value. If the condition is met the record is included in the output file.

Note: This process will drop the entire record. It will not just set the "skip" property to true for later processing. Once the record is dropped, it can no longer be processed by any other process. Multiple identifier and resource definitions are treated as OR conditions. If any one of the conditions is met, the record is included. If a field specified for inclusion contains a blank value, the record is included.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `IncludeRecsByValue` stage appears as follows:

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

The output CSR file appears as follows:

```
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan", 2,EXEMRCV,1,EXBYRCV,2748
>Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan", 2,EXEMRCV,1,EXBYRCV,3013
>Example,20070117,20070117,00:00:00,23:59:59,1,2,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 included.

MaxRecords:

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

Settings

The MaxRecords stage accepts the following input elements.

Attributes:

The following attributes can be set in the <MaxRecords> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Parameters :

The following attributes can be set in the <Parameter> element:

- number = "numeric_value": This parameter allows you to specify the maximum number of records that can be processed.

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

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
```

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

PadIdentifier:

The PadIdentifier stage allows you to pad an identifier with a specified character, either to the left or the right of the identifier.

Settings

The PadIdentifier stage accepts the following input elements:

Attributes:

The following attributes can be set in the PadIdentifier element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Identifiers:

The following attribute can be set in the <Identifier> element:

- name = "identifier_name": This parameter specifies the identifier to be padded.

Parameters:

The following attributes can be set in the <Parameter> element:

- length = "numeric_value" : This parameter specifies the length that the identifier should be.
- padChar = "any_char" : This parameter specifies the pad character to use. The default is 0.
- justify = "left | right" : This parameter specifies left or right justification for the identifier, prior to padding.

Note: Justification specifies how to justify the identifier before executing the padding, therefore justifying the identifier to the right, means the padding will take place to the left of the identifier value.

Example

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

```
WinDisk,20100603,20100603,00:00:00,23:59:59,,3,Feed,Server1-C,Path,C:,Folder,C:,2,DISKFILE,9,
DISKSIZE,3.290056

WinDisk,20100603,20100603,00:00:00,23:59:59,,3,Feed,Server1-C,Path,"C:\Program Files",Folder,
"Program Files",2,DISKFILE,25335,DISKSIZE,4.145787
```

If the PadIdentifier stage appears as follows:

```
<Stage name="PadIdentifier" active="true">
 <Identifiers>
 <Identifier name="Folder"/>
 </Identifiers>
 <Parameters>
 <Parameter length="20"/>
```

```
 <Parameter padChar="?"/>
 <Parameter justify="right"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
WinDisk,20100603,20100603,00:00:00,23:59:59,1,4,Feed,"Server1-C",Path,"C:",Folder,
"????????????????????C:",Account_Code,"????????????????????C:",2,DISKFILE,9,DISKSIZE,
3.290056

WinDisk,20100603,20100603,00:00:00,23:59:59,1,4,Feed,"Server1-C",Path,"C:\Program Files",
Folder,"???????Program Files",Account_Code,"???????Program Files",2,DISKFILE,25335,
DISKSIZE,4.145787
```

The identifier `Folder` has a length of 20 and is padded with the ? character. The justification is set to right so the identifier is right justified and the padding is performed to the left of the identifier.

Prorate:

This process prorates incoming data based on a proration table and a set of proration parameters.

Settings

The `Prorate` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<Prorate>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Files:

The following attributes can be set in the `<File>` element:

- `name = "file_name"`: This is set to the name of the file that contains the proration table.
- `type="prorationtable | exception"`: There are two types of reference file available. Each has its own options. There is no default; therefore, the type and then the format or encoding must be specified.
- `encoding="system | encoding_scheme"`: This option is available only if the `type` is set to `table`. The encoding can be set to conform to the system encoding or it can be set to any of the standard encoding types, e.g., UTF-8.
- `format="CSROutput | CSRPlusOutput"`: This option is available only if the `type` is set to `exception`. The exception file can be in any output format that is supported by 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.

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `IdentifierName`: Name of identifier field to search.
- `IdentifierStart = "numeric_value"`: First position in field to check. Default is 1.
- `IdentifierLength = "numeric_value"`: Number of characters to compare. Default is the entire field.
- `Audit = "true | false"`: Indicates whether or not to write original fields as audit trail. Default is true.
- `AllowNon100Totals = "true | false"`: - Indicates whether or not total proration percentages must equal 100 percent. The default is true.
- `exceptionProcess="true | false"` : If this is set to true and a match is not found, the record will be written to the exception file. If this is set to false (the default) and a match is not found in the conversion table, the identifier will be added to the record with a blank value. The exception file can be in any output format that is supported by 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.

  Note: If this parameter is set to true, do not use a default identifier entry. Records will not be written to the exception file.
- `NewIdentifier = "identifier_name"`: New identifier field name to assign to the updated field. If not specified, the original name will be used.
- `CatchallIdentifier="identifier_name"` : Identifier to be used if there is no match for the identifier field in the proration table. If no value is entered, the default is Catchall. You may specify more than one catchall parameter. If no catchall parameters are specified, catchall processing will not be used.
  - `CatchallPercent="numeric_value"`: Percentage to be used. Default is 100.
  - `CatchallRate="resource_name"`: Rate code to be prorated. Default is all rate codes.

For a detailed example of the Prorate stage, see the *Prorating resources* section.

RenameFields:

The `RenameFields` stage renames specified identifiers and resources.

Settings

The `RenameFields` stage accepts the following input elements.

Attributes:

The following attributes can be set in the `<RenameFields>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Fields:

The following attributes can be set in the `<Field>` element:

- `name = "resource_name | identifier_name"`: Set this to the name of the existing resource or identifier that you would like to rename.

- newName="string_value": Set this to the new resource or identifier name.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the RenameFields stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",UserName,"joe",2,Emails,1,Bytes,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",UserName,"mary",2,Emails,1,Bytes,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",UserName,"joan",2,Emails,1,Bytes,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",UserName,"joan",2,Emails,1,Bytes,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",UserName,"joe",2,Emails,1,Bytes,2817
```

- The field User was renamed UserName.
- The field EXEMRCV was renamed Emails.
- The field EXBYRCV was renamed Bytes.

RenameResourceFromIdentifier:

The RenameResourceFromIdentifier stage allows you to use an identifier value as the value of a resource (rate code).

Settings

The RenameResourceFromIdentifier stage accepts the following input elements:

Attributes:

The following attributes can be set in the <RenameResourceFromIdentifier> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Resources:

The following attribute can be set in the <Resource> element:

- name = "resource_name" : This parameter allows you to specify the resource that will be renamed.

Identifiers:

The following attribute can be set in the <Identifier> element:

- name = "identifier_name" : This parameter specifies the identifier to be used for the renaming.

Parameters :

The following attribute can be set in the <Parameter> element:

- dropIdentifier = "true | false" : The parameter dropIdentifier specifies whether the identifier should be included in the output CSR file or not. If the parameter is set to true, the identifier will be dropped and not included in the output CSR file. If the parameter is set to false, the identifier will be included in the output CSR file.
- renameType = "prefix | suffix | overwrite": The parameter renameType specifies how the resource is renamed. If the parameter is set to prefix, the name of the resource is started with the identifier value. If the parameter is set to suffix, the name of the resource is ended with the identifier value. If the parameter is set to overwrite (default setting), the name of the resource is replaced with the identifier value.

Example

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

```
WinDisk,20100602,20100602,00:00:00,23:59:59,,3,Feed,Server1-C,Path,C:,Folder,C:,2,DISKFILE,
11,DISKSIZE,1.998663

WinDisk,20100602,20100602,00:00:00,23:59:59,,3,Feed,Server1-C,Path,"C:\Program Files",Folder,
"Program Files",2,DISKFILE,16379,DISKSIZE,3.404005
```

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources>
 <Parameters>
 <Parameter dropIdentifier="false"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+20100602201006020 24C:
",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,Feed,"Server1-
C",Path,"C:",Folder,"C:",Account_Code,"C:
",2,C:,11,DISKSIZE,1.998663

"CSR+20100602201006020 24Program Files
",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,Feed,"Server1-
C",Path,"C:\Program Files",Folder,"Program
Files",Account_Code,"Program Files ",2,C:\Program
Files,16379,DISKSIZE,3.404005
```

The resource DISKFILE has been renamed using the identifier Path.

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources>
```

```
 <Parameters>
 <Parameter dropIdentifier="false"/>
 <Parameter renameType="prefix"/>
 </Parameters>
</Stage>
```

The output CSR file appears as follows:

```
"CSR+20100602201006020240: ",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,
Feed,"Server1- C",Path,"C:",Folder,"C:",Account_Code,"C: ",2,C:DISKFILE,11,
DISKSIZE,1.998663
"CSR+201006022010060202 4Program Files ",WinDisk,20100602,20100602,00:00:00,
23:59:59,1,4,Feed,"Server1- C",Path,"C:\Program Files",Folder,"Program Files",
Account_Code,"Program Files ",2,C:\Program FilesDISKFILE,16379,DISKSIZE,3.404005
```

If the RenameResourceFromIdentifier stage appears as follows:

```
<Stage name="RenameResourceFromIdentifier" active="true">
 <Identifiers>
 <Identifier name="Path"/>
 </Identifiers>
 <Resources>
 <Resource name="DISKFILE"/>
 </Resources> <Parameters>
 <Parameter dropIdentifier="false"/>
 <Parameter renameType="suffix"/>
</Parameters> </Stage>
```

The output CSR file appears as follows:

```
"CSR+20100602201006020240: ",WinDisk,20100602,20100602,00:00:00,23:59:59,1,4,
Feed,"Server1- C",Path,"C:",Folder,"C:",Account_Code,"C: ",2,DISKFILEC:,
11,DISKSIZE,1.998663
"CSR+201006022010060202 4Program Files ",WinDisk,20100602,20100602,00:00:00,
23:59:59,1,4,Feed,"Server1- C",Path,"C:\Program Files",Folder,"Program Files",
Account_Code,"Program Files ",2,DISKFILEC:\Program Files,16379,DISKSIZE,3.404005
```

ResourceConversion:

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

Settings

The ResourceConversion stage accepts the following input elements.

Attributes:

The following attributes can be set in the <ResourceConversion> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Resources:

The following attributes can be set in the <Resource> element:

- name = "resource_name": The name of the new resource. You must add the resource to the SmartCloud Cost Management Rate table if it does not already exist in the table.

- `symbol="a-z"`: This allows you to pick a variable which can be used to represent the value of the new resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

FromResource:

The following attributes can be set in the `<FromResource>` element:

- `name = "resource_name"`: The name of an existing resource the value of which will be used to derive a new resource value.
- `symbol="a-z"`: This allows you to pick a variable that will be given the value of the named resource. This can then be used by the `formula` parameter as part of an arithmetic expression. This attribute is restricted to one lowercase letter (a-z).

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `formula = "arithmetic_expression"` : This can be set to any arithmetic expression using the symbols defined in the `Resources` and `FromResources` element.
- `modifyIfExists=="true | false"`: If this parameter is set to "true" and the identifier already exists, the existing identifier value is modified with the specified value. If this is set to false (the default) the existing identifier value is not changed.

Example

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

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,2817
```

If the `ResourceConversion` stage appears as follows:

```
<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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,60,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,60,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,60,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,60,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,60,EXBYRCV,2817
```

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

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.

Settings

The Sort stage accepts the following input elements.

Attributes:

The following attributes can be set in the <Sort> element:

- active = "true | false": Setting this to true activates this stage within a job file. (The default setting is true.)
- trace = "true | false" : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- stopOnStageFailure = "true | false": Setting this to true will halt execution of this stage should an error occur. (The default setting is true)

Identifiers:

The following attributes can be set in the <Identifier> element:

- name = "identifier_name": This allows you to set the identifier on which the sort is based..
- length="numeric value": 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 a length is specified and the length of the field is less than the specified length, blanks will be used to pad out the length.

Parameters:

The following attributes can be set in the <Parameter> element:

- Order="Ascending | Descending": This allows you to set the sort type. The default order is ascending.

Note: This process is memory dependent. If there is not enough memory to do the sort, the process will take a long time to complete.

```
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr1,User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr2,User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,2748
Example,20070117,20070117,00:00:00,23:59:59,,2,Feed,Srvr3,User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,,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,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr2",User,"mary",2,EXEMRCV,1,EXBYRCV,3863
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,2817
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr1",User,"joe",2,EXEMRCV,1,EXBYRCV,3941
Example,20070117,20070117,00:00:00,23:59:59,1,2,Feed,"Srvr3",User,"joan",2,EXEMRCV,1,EXBYRCV,3013
Example,20070117,20070117,00:00:00,23:59:59,1,2,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.

TierResources:

The `TierResources` stage allows tiering of a list of rate codes or all applicable rate codes. A list of resources (rate codes) can be explicitly specified in the XML file. Alternatively, the list of resources (rate codes) can remain empty and the stage determines what resources (rate codes) should be tiered and tiers them.

Overview

Tiered pricing allows you to charge different rates for resource usage based on predefined thresholds. The daily processing of feeds does not change. You continue to process all of your daily feeds as before, with the only change being that the rate values of the proposed tier rates are set to 0. At the end of the accounting period, most commonly a month, a job file can be run to perform the tier pricing for each rate. The basic flow is as follows:

- Extract the summary records for the period using the Universal Collector called DATABASE. The identifiers are `Account_Code` and `RateCode`. The resource must be defined with the name `Units`.
- Run an `Integrator Aggregation` stage to aggregate the `Account_Code`, `RateCode`, and `Units` for each summary record.

  Note: The summary record contains 1 resource only.
  After this stage, a CSR file is created with just `Account_Code`, `RateCode`, and `Units`.
- Run an Integrator `IncludeRecsByValue` stage to drop all of the rates that are not tier rates. You could move this stage before the Aggregation stage if required. After this stage, the CSR file contains just the rate codes that are tier rates.
- Run the Integrator `TierResources` stage. This stage tiers one or more rates at a time.
- Run the standard Bill step to cost the data.
- Run the standard `DBLoad` step to load the costed summary and detail data.

If you decide to change a tier rate, delete the loads for this job and rerun. Set up each tier rate code as described in the requirements section below. It is not important to determine the actual rate value that you want to change, since the reprocessing of this data can be completed by deleting the load with the incorrect data and rerunning.

Requirements

- For resources (rate codes) that you want to tier, you must set the rate values for the parent rates to 0. You do not want to cost the rate during daily processing. The summary records are written to the DB with usage information, but with no cost information.
- You must define the child tier rates with the rate values that you want to charge for in the tier. For example:
  - Set `Z001` parent rate value to 0.

  - Add a child tier rate `Z001_1` with a rate value of `2.00`.
  - Add a child tier rate `Z001_2` with a rate value of 1.8 (10% discount for volume).
  - Add another child tier rate `Z001_3` with a rate value of 1.6.
- The rate pattern used must also be set on the parent tier rate.
- Threshold and Percent values must also be defined on the child tier rates. Percent values are only applicable to rates with the rate pattern "Tier Monetary Individual (%)" and "Tier Monetary Highest (%)".
- A tier rate for the highest threshold must be defined with an empty threshold value, this rate will be used for values above the highest threshold.

Settings

The `TierResources` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<TierResources>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. The default setting is true.
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- `stopOnStageFailure = "true | false"`: Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Resources

The following attributes can be set in the `<Resources>` element:

- `name = "resource_name"`: The name of the resource or rate code used for tiering. Only resources specified under the `<Resources>` element are tiered by this stage. If no resources are specified under the `<Resources>` element, all applicable tiered rates defined in the Rate Tables panel are used.

Example 1: Rate Pattern ="Tier Individual" or "Tier Monetary Individual (%)"

Assume that the following data exists for the rate codes you want to tier.

```
Z001 = 158
Z001 = 300
Z001 = 66
```

And the following thresholds and percentages are defined on the Tiered Rates of Z001:

Table 223. Defined thresholds

RateCode	Threshold	Percentages
Z001_1	100	10
Z001_2	200	15
Z001_3	300	20
Z001_4		25

If the `TierResources` stage appears as follows:

```
<Stage name="TierResources" active="true">
 <Resources>
 <Resource name="Z001"/>
 </Resources>
</Stage>
```

The resource units are tiered as follows, where `Z001_orig` has the original values:

```
Z001_orig,158,Z001_1,100,Z001_2,58
Z001_orig,300,Z001_1,100,Z001_2,100,Z001_3,100
Z001_orig,66,Z001_1,66
```

When using the "Tier Monetary Individual (%)", rate pattern, the Bill step applies the percentage to the resource units at each tier.

Example 2: Rate Pattern ="Tier Highest" or "Tier Monetary Highest (%)"

Assume that the following data exists for the rate codes you want to tier.

```
Z001 = 158
Z001 = 300
Z001 = 66
```

And the following thresholds and percentages are defined on the Tiered Rates of Z001:

Table 224. Defined thresholds

RateCode	Threshold	Percentages
Z001_1	100	10
Z001_2	200	15
Z001_3	300	20
Z001_4		25

If the `TierResources` stage appears as follows:

```
<Stage name="TierResources" active="true">
 <Resources>
 <Resource name="Z001"/>
 </Resources>
</Stage>
```

The resource units are tiered as follows, where `Z001_orig` has the original values:

```
Z001_orig,158,Z001_2,158
Z001_orig,300,Z001_3,300
Z001_orig,66,Z001_1,66
```

When using the "Tier Monetary Highest (%)" rate pattern, the Bill step applies the percentage to the resource units at the corresponding tier.

Additional example

The `<TUAM_home_dir>/samples/jobfiles/SampleTierResources.xml` job file contains an example that shows the process described in this topic.

Related concepts:

Universal Collector overview

The Universal Collector within Integrator was designed to extend the input stage of the integrator and simplify the creation of a new collector.

TierSingleResource:

The `TierSingleResource` stage allows tiering of a single rate code.

Note: This stage has been deprecated and replaced with the new `TierResources` stage. See the related concept topic for more information about the `TierResources` stage. When migrating from this stage to the `TierResources` stage, tiered rates must be redefined manually using the Rate Tables panel. For more information about defining rate tables, see the related *Administering the system guide*.

Overview

Tiered pricing allows you to charge different rates for resource usage based on predefined thresholds. The daily processing of feeds does not change. You continue to process all of your daily feeds as before, with the only change being setting the rate values of the proposed tier rates to 0.

At the end of the accounting period (most commonly a month), a jobfile can be run to perform the tier pricing for each rate. The basic flow is as follow:

- Extract the summary records for the period – using the Integrator Generic Collector – DB. The identifiers will be Account Code and Rate Code. The resource will have the name Units.
- Run an Integrator Aggregation step to aggregate the account code, rate code and units for each summary record. Remember, the summary record only contains 1 resource – so after this step, you will have a CSR file with just account codes, rate codes and units.
- Run an Integrator `IncludeRecsByValue` step to drop all of the rates that are not tier rates. You could move this step before the Aggregation step if you want. After this step, you will have a CSR file with just the rate codes that are tier rates.
- Run one or more Integrator `TierSingleResource` steps. This step will tier one rate at a time. So if you have multiple rates that you want to tier, you must run this step for each rate.
- Run the standard Bill step to cost the data.
- Run the standard DBLoad step to load the costed summary and detail data.

If you decide to change a tier rate, delete the loads for this job and rerun.

Set up each tier rate code as described in the following section. It is not important to determine the actual rate value that you want to change, since the re-processing of this data is easy to do.

Requirements

- Resources (rate codes) that you want to Tier – you must set the rate values for those rates to 0. You do not want to cost the rate during your daily processing. The summary records will be written to the DB with usage information, but with no cost information.
- Resources (rate codes) that you want to Tier, can only be 6 bytes long. If you want to tier a default rate that is longer than 6 bytes, you must rename that resource in an Integrator step. This is because, the process will dynamically add –n to the rate code for each tier. For example, if the tier rate code is Z001, then the tier 1 rate will be Z001-1 and the tier 2 rate will be Z001-2, and so on.
- You must define the tier rates with the rate values that you want to charge for the tier. So again, in the previous example, set Z001 rate value to 0, add a rate

Z001-1 with a rate value of say 2.00 and add rate Z001-2 with a rate value of say 1.8 (10% discount for volume) and add rate Z001-3 with a rate value of 1.6.

Settings

The `TierSingleResource` stage accepts the following input elements:

Attributes:

The following attributes can be set in the `<TierSingleResource>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. (The default setting is true.)
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. (The default setting is false.)
- `stopOnStageFailure = "true | false"`: Setting this to true will halt execution of this stage should an error occur. (The default setting is true.)

Parameters:

The following attributes can be set in the `<Parameter>` element:

- `thresholds = "n {,n}"`: This parameter allows you to specify the threshold values to be used for tiering.
- `ratecode = "rate_code"`: This parameter allows you to specify the rate code to be used for tiering.
- `method = "HighestTier | IndividualTier"`: This parameter allows you to specify the tiering option to be used. With HighestTier, where ever the resource units fall, all of the units will be charged at that rate. With IndividualTier (the default), the units will be charged in their respective tier. For example, if the thresholds are 10,20,30 and the units are 25, then 10 will be charged at tier 1, 10 at tier 2 and 5 at tier 3.

Note: There is always one more tier than number of tiers specified on the thresholds. For example, if you set thresholds 10, 20, 30 – then there are 4 tiers: tier 1 is 0-10, tier 2 is > 10-20, tier 3 is > 20-30 and tier 4 is everything over 30.

Example

Assume that the following data exists for the rate code that you want to Tier:

```
ResourceUnits = 158
ResourceUnits = 300
ResourceUnits = 66
```

If the TierSingleResource stage appears as follows:

```
<Stage name="TierSingleResource" active="true">
 <Parameters>
 <Parameter thresholds="100,200,300"/>
 <Parameter ratecode="ABC101"/>
 <Parameter method="IndividualTier"/>
 </Parameters>
</Stage>
```

The resource units will be tiered as follows:

```
ResourceUnits,158, ABC101-1,100, ABC101-2,58
ResourceUnits,300, ABC101-1,100, ABC101-2,100, ABC101-3,100
ResourceUnits,66, ABC101-1,66
```

If the TierSingleResource stage appears as follows:

```
<Stage name="TierSingleResource" active="true">
 <Parameters>
 <Parameter thresholds="100,200,300"/>
 <Parameter ratecode="ABC101"/>
 <Parameter method="HighestTier"/>
 </Parameters>
</Stage>
```

The resource units will be tiered as follows:

```
ResourceUnits,158, ABC101-2,158
ResourceUnits,300, ABC101-3,300
ResourceUnits,66, ABC101-1,66
```

UpdateConversionFromRecord:

The `UpdateConversionFromRecord` stage inserts and/or updates lookup records in the SmartCloud Cost Management database conversion table based on the process definition for a usage period. The `UpdateConversionFromRecord` stage uses an identifier for the source of the lookup and an identifier for the target of the lookup. Modifications to the conversion records for a process definition only occurs after the `lockdown` date of the process definition.

Settings

The `UpdateConversionFromRecord` stage accepts the following input elements.

Attributes

The following attributes can be set in the `<UpdateConversionFromRecord>` element:

- `active = "true | false"`: Setting this to true activates this stage within a job file. The default setting is true.
- `trace = "true | false"` : Setting this to true enables the output of trace lines for this stage. The default setting is false.
- `stopOnStageFailure = "true | false"`: Setting this to true halts the execution of this stage if an error occurs. The default setting is true.

Identifiers

The following attributes can be set in the `<Identifier>` element:

- `name = "identifier_name"`: The identifier name used as the target for a conversion record in the conversion table. Only one target identifier can be specified for each stage.

FromIdentifier

The following attributes can be set in the `<FromIdentifier>` element:

- `name = "identifier_name"`: The name of the first identifier used as the lookup or low identifier to a conversion record in the conversion table. If a match is found for this identifier in the conversion table for the usage period, then the expiry date of the existing record is set to the period immediately before the usage period start date. A new record is then added to the conversion table where the effective date is set to the usage period start date and the expiry date is set to the default value of `Dec 31, 2199`. If no match is found for this identifier in the conversion table for the usage period, a new record is added to the conversion table where the effective date is set to the usage period start date and the expiry date is set to the default value of `Dec 31, 2199`.

- `name = "identifier_name"`: The name of the second identifier used as the lookup or high identifier to the conversion record in the conversion table. Only used for the addition of new conversion records, for example, `allowNewConversionEntries=true`. If used when conversion updates are allowed, for example, `allowConversionEntryUpdate=forwardOnly`, then the conversion record is written to an exception file.
- `offset="numeric_value"`: This value, in conjunction with length allows you to set how much of the original identifier is used as a search string in the conversion table. This value sets the starting position of the identifier portion you want to use. For example, if you use the whole identifier string, your offset is set to 1. However, if you are selecting a substring of the original identifier that started at the third character, then the offset is set to 3. This is set to 1 by default.
- `length="numeric_value"`: This value, in conjunction with offset allows you to set how much of the original identifier is used as a search string in the conversion table. If you want to use five characters of an identifier, the length is set to 5. This is set to 50 by default.

Files

The following attributes can be set in the `<File>` element:

- `name = "file_name"`: This is set to the name of the exception file.
- `type="exception"`: To allow records to be written to an exception file.
- `format="CSROutput | CSRPlusOutput"`: This option is only available if the `type` is set to `exception`. The exception file can be in any output format that is supported by Integrator. The format is defined by the stage name of the output type. For example, if the stage `CSROutput` or `CSRPlusOutput` is active, the exception file is produced as CSR or CSR+ file.

Parameters

The following attributes can be set in the `<Parameter>` element:

- `process="<process_name>"`: The name of the process definition corresponding to the conversion mappings that you want to insert or update. If this is not set, the `Process Id` of the job file is used as the default.
- `useUsageEndDate="true | false"`: Setting this to true means that the usage period end date is used for lookup of the conversion records whose expiry date is after the usage period start date. Setting this to false means that usage period start date is used for lookup of the conversion records whose expiry date is after the usage period start date. The default setting is false.
- `allowNewConversionEntries="true | false"`: Setting this to true means that if there are no conversion records to update, it inserts a new conversion record into the conversion table. Setting this to false means that it writes the conversion record to an exception file. The default setting is true.
- `allowConversionEntryUpdate= "forwardOnly | none"`: Determines how conversion records are updated. Updates are not allowed if a high identifier is specified. The configuration is as follows:
  - `forwardOnly (default):` Updates can only occur in time order whereby the expiry date of the last conversion record is updated and a new conversion record is added.
  - `none`: Does not allow updates and write the conversion records to an exception file. Not affected if high identifier is specified.

- `strictLockdown="true"`: Setting this to true means that if you try to update or insert a conversion record before the Process Definition lock down date, then the conversion record is written to an exception file. The default setting is true.
- `dayBoundary="true"`: Setting this to true means that the conversion records added to the conversion table will have a start date with a time which is the beginning of that day, i.e. midnight. Setting this to false means that the conversion records added to the conversion table will have a start date with a time which is the start time of the CSR record. The default setting is false.

Example 1: allowNewConversionEntries="true"

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

```
VMWARE,20120314,20120314,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM3",User,"Sean",2,VMCPU,4,VMMEM,2048
```

If the `UpdateConversionFromRecord` stage appears as follows:

```
<Stage name="UpdateConversionFromRecord" active="true">
   <Identifiers>
   <Identifier name="User">
   <FromIdentifiers>
   <FromIdentifier name="VMName" offset="1" length="4"/>
   </FromIdentifiers>
   </Identifier>
   </Identifiers>
   <Parameters>
   <Parameter process="VMWARE"/>
   <Parameter useUsageEndDate="false"/>
   <Parameter allowNewConversionEntries="true"/>
   </Parameters>
</Stage>
```

And the conversion table appears as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
```

The updated conversion table appears as follows:

```
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
'VMWARE','VM3',,'2012-03-14 00:00:00','2199-12-31 23:59:59','Sean'
```

The value for the `VMName` identifier and the start date of the usage period was used to determine the new record with new target mappings in the conversion table for a process definition.

Example 2: allowConversionEntryUpdate="forwardOnly"

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

```
VMWARE,20120111,20120111,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",User,"Paul",2,VMCPU,2,VMMEM,2048
VMWARE,20120116,20120116,00:00:00,23:59:59,,3,HostName,"esx1.example.com",VMName,"VM1",User,"Pat",2,VMCPU,2,VMMEM,2048
```

If the `UpdateConversionFromRecord` stage appears as follows:

```
<Stage name="UpdateConversionFromRecord" active="true">
   <Identifiers>
   <Identifier name="User">
   <FromIdentifiers>
   <FromIdentifier name="VMName" offset="1" length="4"/>
   </FromIdentifiers>
   </Identifier>
   </Identifiers>
   <Parameters>
   <Parameter process="VMWARE"/>
```

```
   <Parameter useUsageEndDate="false"/>
   <Parameter allowConversionEntryUpdate="forwardOnly"/>
   </Parameters>
</Stage>
```

And the conversion table appears as follows:

```
Process Name,Source Identifier Low, Source Identifier High, Effective Date, Expiration Date,
Target Account Code
'VMWARE','VM1',,'2012-01-01 00:00:00','2199-12-31 23:59:59','John'
```

The updated conversion table appears as follows:

```
'VMWARE','VM1',,'2012-01-01 00:00:00','2012-01-10 23:59:59','John'
'VMWARE','VM1',,'2012-01-11 00:00:00','2012-01-15 23:59:59','Paul'
'VMWARE','VM1',,'2012-01-16 00:00:00','2199-12-31 23:59:59','Pat'
```

The value for the `VMName` identifier and the start date of the usage period was used to determine the new expiry date for existing records in the conversion table, as well as new records with new target mappings in the conversion table for a process definition.

Considerations for using UpdateConversionFromRecord

Consider the following when using the `UpdateConversionFromRecord` stage:

- Only one new identifier can be specified in the stage.
- The number of definition entries that you can enter in the conversion table is limited by the memory available to the Integrator.
- The exception file can contain records for input data that was before the `lockdown` date of the process definition or anything else that violates the rules for adding new conversion records.
- The first `<FromIdentifier>` is low identifier and the second is high identifier.
- High identifier cannot be specified for updates to conversion records for the existing identifier in the conversion table.
- Process definition is automatically added to the database if it does not exist.
- The `exceptionProcess` parameter must be turned on for records to be written to the exception file. For more information, see the *Enabling exception processing* topic in the Configuration guide.
- If using multiple `UpdateConversionFromRecord` stages within a job step, then each stage must reference a distinct process definition.

Note: Refer to sample jobfile `SampleAutomatedConversions.xml` when using this stage.

Control statements

This section provides details about control statements for the Acct and Bill programs.

Acct Program Control Statements

This section provides the Acct program control statements.

Control statements for the Acct program can be defined in two locations:

- In the `controlCard` attribute in the job file.
- In the Acct Control file, `AcctCntl.txt`, in the process definition directory that the job file points to.

  Note: The Acct control file is compatible with earlier versions and the use of the file is not recommended. It is recommended that you include control statements in the job file.

Any control statements defined as `controlCard` attributes in the job file will overwrite all control statements in the `AcctCntl.txt` file in the process definition directory.

ACCOUNT CODE CONVERSION {SORT}

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

Format:

```
ACCOUNT CODE CONVERSION
```

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

ACCOUNT FIELD

The ACCOUNT FIELD statement defines the identifier(s) that you want to use to build the account code.

Format:

```
ACCOUNT FIELDn,identifier_name,offset_into_identifier,length
```

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

`offset_into_identifier` = 1-255

`length` = 1-127

Note: Although the maximum number of characters for each account field is 127, the overall length of all account fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

Begin at `ACCOUNT FIELD0` and continue sequentially as needed. The account fields are used (along with `DEFINE FIELD` and `DEFINE MOVEFLD` statements) in account code conversion if account code conversion is enabled. If account code conversion is not enabled, then the account code is built directly from this statement or from the `Account_Code` identifier in the input file records (if present).

Example:

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

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

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

DATE SELECTION

The DATE SELECTION statement defines a date range for records to be processed by CIMSAcct. Records are selected by the end date in the record (either the usage end date or the accounting end date, depending on the record type).

Format:

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

You can use the following values:

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

  or

- One of the following keywords:

  Keyword
  Description

  ****RNDATE**
  Selects records based on the run date

  ****CURDAY**
  Selects records based on the run date and the run date less one day

  ****CURWEK**
  Selects records based on the run week (Sun-Sat)

  ****CURMON**
  Selects records based on the run month

  ****PREDAY**
  Selects records based on the run date, less one (day)

  ****PREWEK**
  Selects records based on the previous week (Sun-Sat)

  ****PREMON**
  Selects records based on the previous month

  CURRENT
  Selects current period from the CIMSCalendar table

  PREVIOUS
  Selects previous period from the CIMSCalendar table

The start and end dates of data processed through SmartCloud Cost Management must reflect the start and end dates of the target billing period. If the billing period runs from January 1 through January 31, any records produced with a date of February 1 (or later) should not be included as part of the processing. In most

cases, this data screening has already been performed, either by SmartCloud Cost Management or through the creation of data inputs reflecting the required date range.

Note that it is rare that you would need to limit the date range of the data used as input to Acct. Bill provides the ability to create billing records reflecting a specific date range using the DATE SELECTION control statement for that program.

Examples:

```
DATE SELECTION 20080601 20080630
DATE SELECTION **PREMON
```

DEFINE FIELD

This statement is used only during account code conversion. Use this statement to specify the offset and length of the account code input field value that you want to use for conversion.

Format:

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

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

`offset` = 1-127 (from the start of the account field or the `Account_Code` identifier value)

`length` = 1-127

Note: Although the maximum number of characters for each define field is 127, the overall length of all define fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

Begin at `DEFINE FIELD0` and continue sequentially as needed.

Example:

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

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

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

DEFINE MOVEFLD

The DEFINE MOVEFLD statement is used only during account code conversion. Use this statement to move all or a portion of the account code input field value to a position in the output account code.

Format:

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

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

`offset` = 1-127 (from the start of the account field or the `Account_Code` identifier value)

`length` = 1-127

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

Note: Although the maximum number of characters for each move field is 127, the overall length of all move fields added together cannot exceed 127 characters. Also, keep in mind that the total output account code, including any literals and move field values, cannot exceed 127 bytes.

Begin at `DEFINE MOVEFLD0` and continue sequentially as needed.

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

Example:

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

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

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

BBC,,FINACCT@0

The resulting account code will be `FINACCTXZ`.

EXCEPTION FILE PROCESSING ON

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

Format:

```
EXCEPTION FILE PROCESSING ON
```

Example:

```
EXCEPTION FILE PROCESSING ON
```

Consideration

- If you enable exception processing, do not include a default account code as the last entry in the account code conversion table (e.g., "`,,DEFAULTCODE`"). If a default account number is used, records will not be written to the exception file.

PRINT ACCOUNT NO-MATCH

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

Format:

```
PRINT ACCOUNT NO-MATCH
```

SHIFT

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

Format:

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

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

`code` = 1-9 (the shift code)

`end time` = the end time of the shift

Rules that apply to shift records:

- Day is defined as the first three letters of the day
- Up to nine shifts per day can be specified
- The preceding end time must always be less than the next end time
- No shift spans midnight

Example:

For the following shifts:

Monday through Friday

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

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

Shift 3 - 5 PM to 8 PM

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

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

Saturday and Sunday

Shift 1 - 8 AM to 5 PM

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

The SHIFT statements are as follows:

```
SHIFT SUN 2 0800 1 1700 2 2400
SHIFT MON 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400
SHIFT TUE 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400
SHIFT WED 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400
SHIFT THU 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400
SHIFT FRI 4 0500 1 0800 2 1130 5 1330 2 1530 1 1700 3 2000 5 2130 4 2400
SHIFT SAT 2 0800 1 1700 2 2400
```

UPPERCASE ACCOUNT FIELDS

The UPPERCASE ACCOUNT FIELDS control statement instructs the Acct program to convert lowercase account code input field values to uppercase values in the resulting account code.

Format:

```
UPPERCASE ACCOUNT FIELDS
```

For example, the value ddic would be converted to DDIC. By using this option, Acct account code processing becomes case-insensitive and makes defining account conversion tables much easier.

Bill Program Control Statements

This section provides documentation on how to define Bill program control statements.

Control statements for the Bill program are defined In the controlCard attribute in the job file.

A sample job file, SampleBillParameters.xml, which illustrates how to use each of the control statements, is added with the installation of SmartCloud Cost Management. SampleBillParameters.xml can be found in the <SCCM_install_dir>\samples\jobfiles directory.

Note: It is possible to define control statements in the Bill Control file, BillCntl.txt, in the process definition directory that the job file points to. However, the Bill control file is compatible with earlier versions and the use of the file is not recommended. It is recommended that you include control statements in the job file. Any control statements defined as controlCard attributes in the job file will overwrite all control statements in the BillCntl.txt file in the process definition directory.

The documentation for each control statement contains an example of how that control statement can be placed inline in a job file.

BACKLOAD DATA

The BACKLOAD DATA control statement instructs the Bill program to ignore the close date. The accounting dates created by Bill are the same as the usage end date regardless of the close date.

Format:

```
BACKLOAD DATA
```

Example:

The following is an example of how the control statement is defined inline in a job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="BACKLOAD DATA"/>
  </Parameters>
 </Bill>
</Step>
```

CLIENT SEARCH ON

The CLIENT SEARCH ON control statement instructs the Bill program to use the account code structure definition to search the CIMSClient table for the rate table associated with a client.

Format:

```
CLIENT SEARCH ON
```

This is useful if you are using differential costing; if you are charging Client A different Rates to Client B. If the CLIENT SEARCH ON control statement is not specified, the STANDARD rate table is used for all clients. Setting the CLIENT SEARCH ON control statement means the Bill program uses the account code to search the CIMSClient table for the client-specific rate table. Bill can do this by matching the entire account code or it can match a sub-section of the account code (This is set using the `DEFINE` control statement. For more information, see DEFINE)

Consider the following example control statements, which are defined inline:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="Client Search On"/>
       <Parameter controlCard="DEFINE J1 1 2"/>
       <Parameter controlCard="DEFINE J2 1 5"/>
  </Parameters>
 </Bill>
</Step>
```

Assume the data value for J1 and J2 is AABBB (J1 = AA, J2 = AABBB). Bill searches the CIMSClient table for account code in its entirety, AABBB. If account

code AABBB is not found, CIMSBill then searches the table for account code AA. If account code AA is found, the rate table for account code AA is used. If account code AA is not found, the STANDARD rate table is used.

DATE SELECTION

The DATE SELECTION control statement defines a date range for records to be processed by the Bill program. Records are selected by the accounting end date in the record.

Format:

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

You can use the following values:

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

  or

- One of the following keywords:

  Keyword
  Description

  ****RNDATE**
  Selects records based on the run date

  ****CURDAY**
  Selects records based on the run date and the run date less one day

  ****CURWEK**
  Selects records based on the run week (Sun-Sat)

  ****CURMON**
  Selects records based on run month

  ****PREDAY**
  Selects records based on run date, less one (day)

  ****PREWEK**
  Selects records based on previous week (Sun-Sat)

  ****PREMON**
  Selects records based on previous month

  CURRENT
  Selects current period from the CIMSCalendar table

  PREVIOUS
  Selects previous period from the CIMSCalendar table

Examples:

Consider the following from and to example control statement, which is defined inline:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
```

```
   <Parameter controlCard="DATE SELECTION 20080601 20080630"/>
  </Parameters>
 </Bill>
</Step>
```

The following inline example control statement demonstrates the use of a date selection keyword:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DATE SELECTION **PREMON"/>
  </Parameters>
 </Bill>
</Step>
```

DEFAULT CLOSE DAY

The DEFAULT CLOSE DAY control statement overrides the system-wide close date set on the Administration Console **System Configuration** > **Cost Management Configuration** > **Processing** page. The year and month used for the close day reflect the year and month in which the Bill program is run.

Format:

```
DEFAULT CLODE DAY=nn
```

`nn` = 01-31

Example:

The following example demonstrates an inline use of the DEFAULT CLOSE DAY control statement:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DEFAULT CLOSE DAY=8"/>
  </Parameters>
 </Bill>
</Step>
```

This statement sets the close date to the eighth of the month. If the Bill run date is 20080706, the close date is set to 20080708. If the Bill run date is 2008712, the close date is also set to 2008708.

DEFINE

The DEFINE statement defines the account code structure used by the statements CLIENT SEARCH ON and DYNAMIC CLIENT ADD ON. This statement also defines character strings in the Detail record that are used for include/exclude processing. The format differs depending on the use of the statement as described in the following sections.

Account Code Fields Define

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

Format:

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

n = 1-9

start_loc = starting position within the account code

len = total length of the level

desc = a description for the field

Example:

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

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DEFINE J1 1 3"/>
   <Parameter controlCard="DEFINE J2 1 6"/>
   <Parameter controlCard="DEFINE J3 1 8"/>
   <Parameter controlCard="DEFINE J4 1 13"/>
   <Parameter controlCard="CLIENT SEARCH ON"/>
  </Parameters>
 </Bill>
</Step>
```

For the previous define fields:

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

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

Include/Exclude Fields Define

This statement defines any string of characters within the Detail record for include/exclude processing.

Format:

```
DEFINE fd loc len /desc/
```

`fd` = two-character field ID, for example A1

`loc` = starting position in the Detail record. For example 163 is the start of the account code field.

`len` = length of the field

`desc` = a description for the field

Example:

In this example, you want to define `UserName` identifier, which starts at position 301 of the Detail file, so that you can exclude all `UserName` values that begin with `geo`.

First, define the field in the Acct program using the `INCLUDE FIELD` statement

```
INCLUDE FIELD0,UserName,1,10
```

Define the field in the Bill program:

```
DEFINE A1 301 3 /usergeorge/
```

And then use the following `EXCLUDE` statement:

```
EXCLUDE A1 geo geo
```

DYNAMIC CLIENT ADD ON

The DYNAMIC CLIENT ADD ON control statement specifies that the Bill program automatically insert corresponding client entries in the CIMSClient table for all clients that have a resource usage charge, but are not entered in the table.

Format:

```
DYNAMIC CLIENT ADD ON
```

Note: This statement must be used with caution because it can quickly fill the CIMSClient table with extraneous records

If you are using multiple rate tables for clients, this statement requires that the control statement `CLIENT SEARCH ON` is also used.

The inserted client entries will include account names only if a matching top level account is currently in the CIMSClient table. For example, if account code AA exists in the CIMSClient table and account code AABBB is inserted, account AABBB is assigned the same account name as AA. However, contacts and budgets assigned to the top level account are not copied to the new client.

Example:

The following example demonstrates how the DYNAMIC CLIENT ADD ON control statement is defined inline:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DYNAMIC CLIENT ADD ON"/>
  </Parameters>
 </Bill>
</Step>
```

EXCLUDE

The EXCLUDE control statement specifies an exclude record condition. The specified data from the field must be equal to or greater than the low value and equal to or less than the high value.

Format:

```
EXCLUDE fd low high
```

`fd` = two-character field id (This is set using the `DEFINE` control statement. For more information, see DEFINE.)

`low` = specifies the low or from selection value (1 to 8 characters)

`high` = specifies the high or to selection value (1 to 8 characters)

Example:

Assume that you want to exclude records with usage start dates between the dates of June 1, 2008 and June 12, 2008. You need to define the field that contains the usage start date in the Detail records. This field begins at position 123.

The following example demonstrates how this is defined inline within your job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DEFINE A1 123 8 /Start Date/"/>
   <Parameter controlCard="EXCLUDE A1 20080601 20080612"/>
  </Parameters>
 </Bill>
</Step>
```

INCLUDE

The INCLUDE control statement specifies an include record condition. The specified data from the field must be equal to or greater than the low value and equal to or less than the high value.

Format:

```
INCLUDE fd low high
```

`fd`: two-character field ID (This is set using the `DEFINE` control statement. For more information, see "DEFINE" on page 582)

`low`: specifies the low or from selection value (1 to 8 characters)

`high`: Specifies the high or to selection value (1 to 8 characters)

Example:

Assume that you want to include records with usage start dates between the dates of June 1, 2008 and June 12, 2008. You need to define the field that contains the usage start date in the Detail records. This field begins at position 123.

The following example demonstrates how this is defined inline within your job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="DEFINE A1 123 8 /Start Date/"/>
   <Parameter controlCard="INCLUDE A1 20080601 20080612"/>
  </Parameters>
 </Bill>
</Step>
```

KEEP ORIGINAL CPU VALUES

If CPU normalization is performed, the KEEP ORIGINAL CPU VALUES control statement instructs the Bill program to save the original CPU value as an identifier value.

Format:

```
KEEP ORIGINAL CPU VALUES
```

The identifier name is prefixed by `Orig_` followed by the rate code.

Example:

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

The following example demonstrates how this is defined inline within your job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="KEEP ORIGINAL CPU VALUES"/>
  </Parameters>
 </Bill>
</Step>
```

NORMALIZE CPU VALUES

This statement instructs the Bill program to normalize CPU values.

Format:

```
NORMALIZE CPU VALUES
```

Example:

The following example demonstrates how this control statement is defined inline in your job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="NORMALIZE CPU VALUES"/>
  </Parameters>
 </Bill>
</Step>
```

REPORT DATE

The REPORT DATE statement specifies the dates that are used as the accounting dates in the Summary records created by the Bill program. This is the date that is used for reporting purposes in SmartCloud Cost Management.

For more information on how accounting dates are derived and set, see the section *Setting accounting dates*.

Note: The use of this statement is not recommended. This statement will place report dates rather than actual usage end dates in the accounting date fields of the Summary records.

Format:

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

You can use the following values:

- From and to dates

  or

- One of the following keywords:

  Keyword
  Description

****RNDATE**
Sets the date range based on the run date

****CURDAY**
Sets the date range based on the run date and the run date less one day

****CURWEK**
Sets the date range based on the run week (Sun-Sat)

****CURMON**
Sets the data range based on the run month

****PREDAY**
Sets the date range based on the run date, less one day

****PREWEK**
Sets the date range based on the previous week (Sun-Sat)

****PREMON**
Sets the date range based on the previous month

CURRENT
Sets the date range based on the current period from the CIMSCalendar table

PREVIOUS
Sets the date range based on the previous period from the CIMSCalendar table

USE DETAIL DATES
Sets the date range based on the Resource record (consult IBM before using this keyword)

Examples:

The following example demonstrates how this control statement is defined using a date range:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="REPORT DATE 20080601 20080630"/>
  </Parameters>
 </Bill>
</Step>
```

The following example demonstrates how this control statement is defined using a keyword:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="REPORT DATE **PREMON"/>
  </Parameters>
 </Bill>
</Step>
```

```
REPORT DATE **PREMON
```

USE SHIFT CODES

The USE SHIFT CODES control statement instructs the Bill program to use the shift character specified in the Resource record and apply the appropriate rate shift value.

Format:

```
USE SHIFT CODES
```

If a shift value is not defined for the rate, the default rate value for the resource will be used. The default rate value is also the SHIFT1 rate value.

Example:

The following example demonstrates how this control statement is defined inline in a job file:

```
<Step   id="Bill"
 description="Bill step"
 type="Process"
 programName="Bill"
 programType="java"
 active="true">
 <Bill>
  <Parameters>
   <Parameter controlCard="USE SHIFT CODES"/>
  </Parameters>
 </Bill>
</Step>
```

Reports

SmartCloud Cost Management produces chargeback and resource accounting reports based on IT usage data from your organization. To help you to easily create reports that display the information that you need, SmartCloud Cost Management includes a variety of standard reports that you can use as templates.

Cognos Reports

Cognos Reports are located within Tivoli Common Reporting and are named after the report itself. There are no files for the Cognos reports because all actions on the reports are performed within Cognos. It is possible to export the report definition as XML if required. For more information about exporting a report, see Report Studio Professional Authoring User Guide available by clicking F1 from the Report Studio.

Date Ranges

The reports can be run for various date ranges including both Calendar and Fiscal Calendar ranges. The following table describes the date ranges and in which reporting tool they are available.

Table 225. Date Ranges

Date Option	Web Reporting	Cognos based Tivoli Common Reporting	Description
All	Yes	Yes	All dates

Table 225. Date Ranges (continued)

Date Option	Web Reporting	Cognos based Tivoli Common Reporting	Description
Date Range Below / Custom	Yes	Yes	A user defined date range
Today	Yes	Yes	Today
Yesterday	Yes	Yes	Yesterday
Last 7 days	No	Yes	Previous 7 days including current date
Last 30 days	No	Yes	Previous 30 days including current date
Last 90 days	No	Yes	Previous 90 days including current date
Last 365 days	No	Yes	Previous 365 days including current date
Current Week to date	Yes	Yes	The start of the calendar week to the current date
Current Month to date	Yes	Yes	The start of the calendar month to the current date
Current Year to date	Yes	Yes	The start of the calendar year to the current date
Last Week	Yes	Yes	The previous calendar week
Last Month	Yes	Yes	The previous calendar month
Last year	Yes	Yes	The previous calendar year
Current Week	Yes	Yes	The current calendar week
Current Month	Yes	Yes	The current calendar month
Current Year	Yes	Yes	The current calendar year
Current Period	Yes	Yes	The current fiscal period as defined in the `CIMSCalendar` table
Previous Period	Yes	Yes	The previous fiscal period as defined in the `CIMSCalendar` table
Fiscal Year	Yes	Yes	The fiscal year as defined in the `CIMSCalendar` table
Previous Fiscal Year	Yes	Yes	The previous fiscal year as defined in the `CIMSCalendar` table
Fiscal Year to date	Yes	Yes	The start of the fiscal year as defined in the `CIMSCalendar` table to the current date
Previous Fiscal Year to date	Yes	Yes	The previous fiscal year as defined in the `CIMSCalendar` table to the current date of the previous year

The reports support a fiscal year with either 12 or 13 periods.

Note: The start of a calendar week is Sunday.

Cognos based Tivoli Common Reporting reports

Cognos based Tivoli Common Reporting requires no knowledge of the database or SQL to create reports. All data is available to you to ensure you can find the information you need quickly. Reports can be published to a public or private area for use.

The Cognos reports in IBM SmartCloud Cost Management are grouped together in folders. For example, accounting reports are located in the **Account Reports** folder, budget reports in the **Budget Reports** folder, and so on. Information such as parameters, tables, output and usage for each report is described in the following topics.

Note: For more information about the Cognos reports mentioned in this section and other best practices, see the IBM SmartCloud Cost Management wiki: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/IBM%20SmartCloud%20Cost%20Management/page/Welcome

Account reports

The **Account reports** folder is used to group all the Cognos Accounting reports in one location. Accounting reports are used to show account level information for usage and charge.

Account Summary YTD report:

The Account Summary YTD report provides the total period and YTD charges by account code, rate group, and rate description for the parameters selected.

Table 226. Account Summary YTD details

Name	Account Summary YTD report
Purpose	Use this report to monitor the breakdown of charges by rate group and a breakdown of charges by rate description.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Year` – The accounting year that the report is run for. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level. **Note:** Selecting both `Show Consolidated` and `Show Child Rate` shows non-tiered and tiered rates for both the parent and child rate. If you choose not to select these two options, the parent and child rates display non-tiered rates only.

Table 226. Account Summary YTD details (continued)

Name	Account Summary YTD report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT – Account contact data • SCDIMRATE – Rate data. • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows the total period and YTD charges by account code, rate group, and rate description for the parameters selected. You can drill down to see a breakdown of charges by rate group and a breakdown of charges by rate description.
Usage	The report enables you to monitor and understand the distribution of the YTD charges by account code, rate group, and rate description for the parameters selected.

Account Total Invoice report:

The Account Total Invoice report provides the total charges for each level of the account code structure for the parameters selected.

Table 227. Account Total Invoice report details

Name	Account Total Invoice report
Purpose	Use this report to monitor the charges at each level of the account code structure
Tivoli Common Reporting folder	Account Reports

Table 227. Account Total Invoice report details (continued)

Name	Account Total Invoice report
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows charges at each level of the account code structure and allows users to drill down and see the charges at each level.
Usage	The report enables you to monitor and understand the distribution of the charges within an account code structure.

Application Cost report:

The Application Cost report provides the total charges for up to four levels of an account code structure. The report shows the usage, rate value and charge by `rategroup` and rate for each account level up to the lowest account level chosen.

Table 228. Application Cost report details

Name	Application Cost report
Purpose	Use this report to monitor the charges within an account code structure and drill down to see the resources they are using.
Tivoli Common Reporting folder	Account Reports

Table 228. Application Cost report details (continued)

Name	Application Cost report
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level. **Note:** Selecting both `Show Consolidated` and `Show Child Rate` shows non-tiered and tiered rates for both the parent and child rate. If you choose not to select these two options, the parent and child rates display non-tiered rates only.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSDIMUSERRESTRICT` – User security data • `SCDIMRATE` – Rate data • `CIMSCALENDAR` – Financial calendar data • `CIMSCONFIGOPTIONS` – Configuration data • `CIMSCONFIG` – Configuration data
Output	The report shows the charges at each level of the account code structure up to the level chosen. You can drill down to see details of the usage and charges for the corresponding rate groups and rates.

Table 228. Application Cost report details (continued)

Name	Application Cost report
Usage	The report enables you to monitor and understand the distribution of the charges by account and account level. The report also provides more information about the resources that the accounts are using.

Daily Charges - Charges report:

The Daily Crosstab Charges report provides total charge for a defined period by account code and rate code within rate group by day for the parameters selected.

Table 229. Daily Charges - Charges report details

Name	Daily Charges - Charges report
Purpose	Use this report to monitor the charges for a period broken down by day.
Tivoli Common Reporting folder	Account Reports.
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.

Table 229. Daily Charges - Charges report details (continued)

Name	Daily Charges - Charges report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows charges as a cross tab with the account and rate as the x-axis and the accounting date as the y-axis.
Usage	The report allows you to monitor the charges by day so you can view the trends in charges over time.

Daily Crosstab - Usage report:

The Daily Crosstab Usage report shows the total usage for a defined period by account code and rate code within rate group by day for the parameters selected.

Table 230. Daily Crosstab - Usage report details

Name	Daily Crosstab - Usage report
Purpose	Use this report to monitor the usage for a period broken down by day.
Tivoli Common Reporting folder	Account Reports.

Table 230. Daily Crosstab - Usage report details (continued)

Name	Daily Crosstab - Usage report
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows usage as a crosstab with the account and rate as the x-axis and the accounting date as the y-axis.
Usage	The report allows you to monitor the usage by day so you can view the trends in usage over time.

Monthly Crosstab - Charges report:

The Monthly Crosstab Charges report provides total monthly charge by account code and rate code description for the parameters selected.

Table 231. Monthly Crosstab - Charges report details

Name	Monthly Crosstab - Charges report
Purpose	Use this report to monitor the charges by account and rate.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows charges as a crosstab with the account and rate on the x-axis and the period as the y-axis.
Usage	The report enables you to monitor the charges by period so you can view the trends in charges over time.

Monthly Crosstab - Usage report:

The Monthly Crosstab Usage report provides total monthly resource usage by account code and rate code description for the parameters selected.

Table 232. Monthly Crosstab Usage details

Name	Monthly Crosstab Usage report
Purpose	Use this report to monitor the total monthly resource usage by account code and rate code description for the parameters selected.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSDIMCLIENTCONTACT` – Account contact data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows the total monthly resource usage by account code and rate code description for the parameters selected.

Table 232. Monthly Crosstab Usage details (continued)

Name	Monthly Crosstab Usage report
Usage	The report enables you to monitor monthly resource usage by account code and rate code description for the parameters selected.

Percentage report:

The Percentage report provides the total charge by account code for the parameters selected. The report specifies the percentage of the charge in relationship to the total charges for all account codes. This report also provides a breakdown of the percentage by rate group and rate code description for each account code.

Table 233. Percentage report details

Name	Percentage report
Purpose	Use this report to monitor the charges for an account and understand which services the budget is being spent on.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.

Table 233. Percentage report details (continued)

Name	Percentage report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows the charges by account as a pie chart and a table showing the percentages at account, rate group and rate code level.
Usage	The report allows you to understand how an account spends its budget.

Summary Crosstab - Charges report:

The Summary Crosstab Charges report provides total charges for a defined period by account code and rate code within rate group for the parameters selected.

Table 234. Summary Crosstab - Charges report details

Name	Summary Crosstab - Charges report
Purpose	Use this report to monitor the charges for a period.
Tivoli Common Reporting folder	Account Reports

Table 234. Summary Crosstab - Charges report details (continued)

Name	Summary Crosstab - Charges report
Parameters	• `Account Structure` – The account code structure is used for the account codes in reporting. • `Account Code Level` – The level within the account code structure to display the account codes at. • `Starting Account Code` – The lower boundary denoting the account codes for the report to show. • `Ending Account Code` – The upper boundary denoting the account codes for the report to show. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Rate Group` – The rate group to show the rate details for in the report (optional). • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows charges as a crosstab with the Account as the x-axis and the rate group and rate as the y-axis.
Usage	The report allows you to monitor the charges by period so you can view the trends in charges over time.

Summary Crosstab - Usage report:

The Summary Crosstab Usage report provides total usage for a defined period by account code and rate code within a rate group for the parameters selected.

Table 235. Summary Crosstab - Usage report details

Name	Summary Crosstab - Usage report
Purpose	Use this report to monitor the usage for a period.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes at. • `Starting Account Code` – The lower boundary, denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Group` – The rate group to show the rate details for in the report (optional). • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows usage as a crosstab with the account as the y-axis and the rate group and rate as the x-axis.
Usage	The report allows you to monitor the usage by period so you can view the trends in usage over time.

Weekly Crosstab - Charges report:

The Weekly Crosstab Charges report provides total charges for a defined period by account code and rate code within rate group by week for the parameters selected.

Note: The accounting week starts on a Sunday and is constrained by monthly boundaries. If the start of the week for a record is in the previous month, then the week start date for that record will be the first day of the month.

Table 236. Weekly Crosstab - Charges report details

Name	**Weekly Crosstab - Charges report**
Purpose	Use this report to monitor the charges for a period broken down by week.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows charges as a crosstab with the account and rate as the x-axis and the accounting week as the y-axis.

Table 236. Weekly Crosstab - Charges report details (continued)

Name	Weekly Crosstab - Charges report
Usage	The report allows you to monitor the usage by day so you can view the trends in usage over time.

Weekly Crosstab - Usage report:

The Weekly Crosstab Usage report provides total usage for a defined period by account code and rate code within rate group by week for the parameters selected.

Note: The accounting week starts on a Sunday and is constrained by monthly boundaries. If the start of the week for a record is in the previous month, then the week start date for that record will be the first day of the month.

Table 237. Weekly Crosstab - Usage report details

Name	Weekly Crosstab - Usage report
Purpose	Use this report to monitor the usage for a period broken down by week.
Tivoli Common Reporting folder	Account Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.

Table 237. Weekly Crosstab - Usage report details (continued)

Name	Weekly Crosstab - Usage report
Tables/Views Used	• SCSSUMMARY – Summary data • CIMSDIMCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows usage as a crosstab with the account and rate as the x-axis and the accounting week as the y-axis.
Usage	The report allows you to monitor the usage by week so you can view the trends in usage over time.

Budget reports

The **Budget reports** folder is used to group all the Cognos Budget reports in one location. Budget reports are used to compare the actual and budget charges for accounts and resources.

Client Budget report:

The Client Budget report shows the budget and the actual charges for account codes. You can also view the difference for a selected period and at year-to-date level. In addition, a bar chart is displayed showing the total budget and actuals for all accounts by period up to the selected period for the selected year.

Table 238. Client Budget report details

Name	Client Budget report
Purpose	Use this report to understand how charges for accounts are matching their budgets.
Tivoli Common Reporting folder	Budget Reports
Parameters	• Account Structure – The account code structure that is used for the account codes in the reporting. • Account Code Level – The level within the account code structure that is used to display the account codes. • Starting Account Code – The lower boundary denoting the account codes displayed in the report. • Ending Account Code – The upper boundary denoting the account codes displayed in the report. • Date Range – The list of predefined periods to run the report for. • Year – The year to run the report for. • Period – The period to run the report for.

Table 238. Client Budget report details (continued)

Name	Client Budget report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSCLIENTBUDGET – Client budget data • CIMSDIMCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows a bar chart with the total actual and budget values for all accounts selected by period. In addition, a table showing the budget and charges for the current period and year-to-date is also shown. For those accounts where the charges accrued are greater than the assigned budget, the row is shown in red.
Usage	The report allows you to understand how an account is keeping to its budget.

Line Item Budget report:

The Line Item Budget report provides actual, budget, and difference charges by account code, rate group, and rate code description for the parameters selected. This report shows totals for the calendar period selected and a year to date (YTD) figure. It reflects the amount for the individual resource budgets for the account code defined in the Client budgets.

Table 239. Line Item Budget report details

Name	Line Item Budget report
Purpose	Use this report to see how accounts are accruing charges compared to the defined budget.
Tivoli Common Reporting folder	Budget Reports

Table 239. Line Item Budget report details (continued)

Name	Line Item Budget report
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table for the actual figures. • `Year` – The year to run the report for. • `Period` – The period to run the report for.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSCLIENTBUDGET` – Client budget data. • `CIMSDIMCLIENT` – Account data • `CIMSDIMCLIENTCONTACT` – Account contact data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows for each account code, the total actual, budget, and difference charges for both the period and YTD broken down by rate within rate group.
Usage	You can monitor how clients are using resources compared to the defined budget.

Cloud reports

The **Cloud reports** folder contains reports for IBM SmartCloud Cost Management users that use IBM Cloud Orchestrator.

Project Summary report:

The Project Summary report is an active report that shows details of virtual machine usage by Project within domain, which is broken down by user, region, and flavor for the last 7 days, 6 weeks, and 12 periods.

Note: You must install Tivoli Common Reporting 3.1.0.1 to view this report.

Note: As this report shows a large amount of data and may take some time to run, it is recommended to schedule the report.

Table 240. Project Summary report details

Name	**Project Summary report**
Tivoli Common Reporting folder	Cloud reports
Purpose	Use this report to understand virtual machine usage by project.
Parameters	None
Tables/Views Used	• `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier data • `CIMSCONFIG` – Configuration data • `CIMSCONFIGOPTIONS` – Configuration data • `SCDIMCALENDAR` – Financial calendar data

Table 240. Project Summary report details (continued)

Name	Project Summary report
Output	The report shows the following 4 tabs: **Summary** Used to view existing, created, and deleted virtual machines for the period that is selected. The Summary page consists of: • **All Projects:** – Graph view – Shows the number of new, existing, and deleted virtual machines for the selected period. – Table view – Shows the same data as the stacked column bar chart, but in a table view. • **Summary:** Shows the number of domains, projects, users, and virtual machines by flavor and by region for the selected period in a table format. Select from the following options to view the data for a particular period: • Previous 7 days • Previous 6 weeks • Previous 12 periods **Counts** Used to view the number of virtual machines active in the last 7 days. The Counts tab consists of: • • **Summary** view: For a selected project, a summary view is displayed that shows the following details for the last 7 days: – **Region**: the number of distinct regions where the project or users virtual machines are stored. – **Availability Zone**: the number of distinct availability zones where the project or users virtual machines are stored. – **Pattern**: the number of distinct IBM Cloud Orchestrator patterns that are used by the project or users virtual machines. – **Architecture**: the number of distinct architectures that are used by the project or users virtual machines. – **Flavor**: the number of distinct virtual machines owned by the project or users, which are broken down by flavor. **Note:** Flavors of the format `Instance_UUID` are shown as **Other**. Refer to the related Product Limitations topic for more details. – **Virtual Machine Count**: the total number of virtual machines. • **Detail** view: The detail view shows the name information for a selected project, where the counts are displayed broken down by user. Click a project in the **Select Project** list box to view the details for that project. The list box can be hidden or displayed using the **Hide Sidebar** or **Show Sidebar** toggle option. The summary and detail views can be selected

Related concepts:

Product limitations

Dashboard reports

The **Dashboard reports** folder is used to group all the Cognos Dashboard reports in one location. The reports provided in this folder can be used as examples of dashboards in Tivoli Common Reporting.

Top N Account Charges report:

The Top N Account Charges report provides the account codes with the highest charges for the parameters selected as a list.

Table 241. Top N Account Charges report details

Name	**Top n Account Charges report**
Purpose	Use this report to see a list of the top N accounts with the highest charges.
Tivoli Common Reporting folder	Dashboard Reports / Individual Dashboard Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Top N` – The number of account codes to show.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a list of the account codes with the highest charges.
Usage	You can regularly monitor the clients defined in the system to see which are accruing the most charge.

Top N Account Charges Pie Chart report:

The Top N Account Charges Pie Chart report provides the account codes with the highest charges for the parameters selected as a pie chart.

Table 242. Top N Account Charges Pie Chart report details

Name	**Top N Account Charges Pie Chart report**
Purpose	Use this report to see a pie chart showing the percentage distribution of charges for the top N accounts with the highest charges.
Tivoli Common Reporting folder	Dashboard Reports / Individual Dashboard Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Top N` – The number of account codes to show.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a pie chart of the account codes with the highest charges. **Note:** The pie chart will be automatically resized depending on the values supplied for Top N and the Account Level Chosen. This may result in a small pie chart being displayed. This is caused by a restriction on the size of the chart on the page so it can be displayed as a dashboard.
Usage	You can monitor the clients defined in the system to see which are accruing the most charge.

Top N Rate Group and Rate Resource Usage report:

The Top N Rate Group and Rate Resource Usage report shows a pie chart and tables for both the top N rate groups and rate codes with the highest % increase in charges between the previous two full periods.

Table 243. Top N Rate Group and Rate Resource Usage report details

Name	**Top N Rate Group and Rate Resource Usage report**
Purpose	Use this report to see the rate groups and rate codes with the greatest increase in charges between the previous two periods to get an understanding of those resource areas that are being used more.
Tivoli Common Reporting folder	Dashboard Reports / Page Dashboard Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Top N` – The number of account codes to show. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `SCCIMSDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a pie chart and table of the rate codes and rate groups with the greatest increase in charges between the previous two periods in a dashboard format.
Usage	You can monitor the resources that are increasing in usage.

Invoice reports

The **Invoice reports** folder is used to group all the Cognos Invoice reports in one location. Invoice reports are used to show charges by account and resource.

Invoice by Account Level report:

The Invoice by Account Level report provides charges by account code, rate group, and rate description for the parameters selected. An optional graph is included showing total expenses by account code.

Table 244. Invoice by Account Level report details

Name	**Invoice by Account Level report**
Purpose	Use this report to invoice accounts for their usage.
Tivoli Common Reporting folder	Invoices

Table 244. Invoice by Account Level report details (continued)

Name	Invoice by Account Level report
Parameters	• Account Structure – The account code structure used for the account codes in the reporting. • Account Code Level – The level within the account code structure to display the account codes at. • Starting Account Code – The lower boundary denoting the account codes displayed in the report. • Ending Account Code – The upper boundary denoting the account codes displayed in the report. • Date Range – The list of predefined periods to run the report for. • Start Date – The lower boundary of the date period used. • End Date – The upper boundary of the date period used. • Rate Table – Used to specify the rate table. • Rate Pattern – Used to specify the rate pattern type. • Show Rate Group – Select this option to see the subtotal in the Rate Group. If you choose not to select this option, the Alternate Invoice report is displayed. • Show Consolidated – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • Show Child Rate – Select this option if you want to display non-tiered rates and tiered rates at the child level. **Note:** Selecting both Show Consolidated and Show Child Rate shows non-tiered and tiered rates for both the parent and child rate. If you choose not to select these two options, the parent and child rates display non-tiered rates only. • Show Graph – Select this option if you want to see a graph showing total expenses by account code. • Invoice Number – Starting Invoice Number to use.

Table 244. Invoice by Account Level report details (continued)

Name	Invoice by Account Level report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT – Account contact data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report optionally shows a graph displaying the total expenses by account code and an invoice for each account. The usage and charges are broken down by rate with an optional subtotal for the rate group. The report totals are shown on the last page. On the graph page, you can navigate from the graph to the relevant invoice page for an account. On the Invoice page, you can drilldown to the same information for the sub-account level by clicking on the account code on a page. Drilldown to the Invoice Detail by RateGroup and Date report is also available by clicking on the rate group name in the rate group sub-total. The Invoice Detail Line Item Resource Units by Identifier report can be drilled into by right-mouse-clicking on the Resource Units value. The Usage by identifier report can also be drilled into by right-mouse-clicking on the Resource Units value. The Charges by Identifier report can be drilled into by clicking on the monetary value. **Note:** The Charges by Identifier and Usage by Identifier reports can only be drilled to from normal rate codes so they are not accessible for tiered rates.
Usage	The report can be used as an invoice for each account.

Invoice Detail Line Item Resource Units by Identifiers report:

The Invoice Detail Line Item Resource Units by Identifiers report enables drill down of resource units by up to 5 identifiers. It is accessed using the Invoice by Account Report by clicking on the units for a rate on the Invoice. Note that this report is hidden and can only be run by using the Invoice by Account report.

Table 245. Invoice Detail Line Item Resource Units by Identifiers report details

Name	**Invoice Detail Line Item Resource Units by Identifiers report**
Purpose	Use this report to understand the detail data for the account shown in the invoice.
Tivoli Common Reporting folder	Invoices / Invoice Drilldowns
Parameters	• `Identifier` – The identifiers used to generate the data. • `Rate Table` – The Rate Table to show the data for (optional).
Tables/Views Used	• `SCDETAIL` – Detail data • `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier types • `SCDIMRATE` – Rate data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows the total units for the selected account and rate from the Invoice by Account Level report, broken down by the selected identifiers.
Usage	The report is used to understand the detailed data for usage for the account shown in the invoice.

Invoice Drill Down for Rate Group by Date report:

The Invoice Drill Down for Rate Group by Date report shows a break down of the usage by identifer and rate for a specified account and rate group. It is accessed using the Invoice by Account report by clicking on the rate group on the Invoice. Note that this report is hidden and can only be run using the Invoice by Account report.

Table 246. Invoice Drill Down for Rate Group by Date report details

Name	**Invoice Drill Down for Rate Group by Date report**
Purpose	Use this report to understand the detail data for the account shown in the invoice.
Tivoli Common Reporting folder	Invoices / Invoice Drilldowns

Table 246. Invoice Drill Down for Rate Group by Date report details (continued)

Name	Invoice Drill Down for Rate Group by Date report
Parameters	• `Identifier` – The identifier to show the data for. • `Date Drilldown` – Used show the resource usage date in the report. • `Rate Table` – The Rate Table to show the data for (optional).
Tables/Views Used	• `SCDETAIL` – Detail data • `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier types • `SCDIMRATE` – Rate data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows the total units as a cross tab with the identifier as the x-axis and the rate codes for the rate group as the y-axis.
Usage	The report is used to understand the detailed data for usage for the account shown in the invoice.

Run Total Invoice report:

The Run Total Invoice report provides total charges by account code within rate description and rate group for the parameters selected.

Table 247. Run Total Invoice details

Name	Run Total Invoice report
Purpose	Use this report to monitor the total charges by account code within rate description and rate group for the parameters selected.
Tivoli Common Reporting folder	Invoices

Table 247. Run Total Invoice details (continued)

Name	Run Total Invoice report
Parameters	• Account Structure – The account code structure that is used for the account codes in the reporting. • Account Code Level – The level within the account code structure to display the account codes. • Starting Account Code – The lower boundary denoting the account codes displayed in the report. • Ending Account Code – The upper boundary denoting the account codes displayed in the report. • Date Range – The list of predefined periods to run the report for. • Start Date – The lower boundary of the date period that is used. • End Date – The upper boundary of the date period that is used. • Rate Pattern – Used to specify the rate pattern type. • Show Consolidated – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • Show Child Rate – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows the total charges by account code within rate description and rate group for the parameters selected. Click the expand icon for a rate description and a breakdown of data by account code is displayed. When Show Consolidated is selected and a tiered parent rate is displayed, click on the parent rate to drilldown to the details for the child rates.
Usage	The report enables you to monitor the total charges by account code within rate description and rate group for the parameters selected.

Run Total Rate Group Percent report:

This report provides charges and percentage for account codes within rate groups for the parameters selected.

Table 248. Run Total Rate Group Percent details

Name	Run Total Rate Group Percent report
Purpose	Use this report to monitor the charges and percentage by rate groups for the parameters selected.
Tivoli Common Reporting folder	Invoices
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period that is used. • `End Date` – The upper boundary of the date period that is used. • `Rate Table` – Used to specify the rate table. • `Rate Group` – The rate group used to show the rate details (optional).
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows the charges and percentage for account codes within rate groups for the parameters selected. Click on the expand icon next to the rate description and a breakdown of the data by account code is displayed.
Usage	The report enables you to monitor the charges and percentage by rate groups for the parameters selected.

Other reports

The **Other reports** folder is used to group all the Cognos Other reports in one location. Other reports are used to show configuration information, including resources and accounts.

Client report:

The Client report provides the information contained in the Client tables for the parameters selected.

Table 249. Client report details

Name	Detail by Identifier report
Purpose	Use this report to monitor the information contained in the Client tables for the parameters selected.
Tivoli Common Reporting folder	Other
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report.
Tables/Views Used	• `CIMSDIMCLIENT` – Account data • `CIMSDIMCLIENTCONTACTNUMBER` – Account contact data • `CIMSDIMCLIENTCONTACT` – Account contact data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data
Output	The report provides the information contained in the Client tables for the parameters selected.
Usage	The report enables you to monitor the information contained in the Client tables for the parameters selected.

Configuration report:

The Configuration report provides information contained in the Configuration tables.

Table 250. Configuration report details

Name	**Configuration report**
Purpose	Use this report to monitor configuration settings.
Tivoli Common Reporting folder	Other
Parameters	None
Tables/Views Used	• `CIMSCONFIG` – Configuration data. • `CIMSCONFIGOPTIONS` – Configuration options data.
Output	The report provides the information contained in the Configuration table.
Usage	Use the report to monitor configuration settings in SmartCloud Cost Management.

Rate report:

The Rate report provides the information contained in the Rate tables.

Table 251. Rate report details

Name	**Rate report**
Purpose	The Rate report provides the information contained in the Rate tables.
Tivoli Common Reporting folder	Other
Parameters	• `Rate Pattern` – The Rate Pattern to show data for (Optional). • `Rate Table` – Used to specify the rate table. • `Rate Group` – The Rate Group to show the rate details for (Optional).
Tables/Views Used	• `SCDIMRATESHIFT` – Rate data.
Output	The report provides the information contained in the Rate tables.
Usage	The report enables you to monitor the information contained in the Rate tables.

Resource Detail reports

The **Resource Detail reports** folder is used to group all the Cognos Resource Detail reports in one location. Resource Detail reports are used to understand the detail data using the identifiers loaded.

Batch report:

The Batch report provides z/OS® batch job data for the parameters selected.

Table 252. Batch report details

Name	Batch report
Tivoli Common Reporting folder	Resource Detail
Purpose	Use this report to see the usage broken down for each z/OS® batch job, subsystem and account for the following rate codes: • Z001 Mainframe Jobs Started • Z002 Mainframe Steps Started • Z003 Mainframe CPU Minutes • Z005 Mainframe Total SIOs • Z006 Mainframe Disk SIOs • Z007 Mainframe Tape SIOs • Z033 Mainframe CPU Minutes (All)
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used.
Tables/Views Used	• `SCDETAIL` – Detail data • `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier types • `CIMSCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a list of the z/OS® batch job, subsystem and account codes with the corresponding usage for each of the above rate codes.
Usage	You can monitor the usage for the z/OS® batch jobs.

Charges by Identifier report:

The Charges by Identifier report shows charges by identifier value for a specified identifier type. The available identifiers depend on the data that is loaded. Available identifiers can include, for example, Virtual Image Name, VM Name, and User.

Table 253. Charges by Identifier report details

Name	**Charges by Identifier report**
Purpose	Use this report to show detailed charges by any provided identifier.
Tivoli Common Reporting folder	Resource Detail

Table 253. Charges by Identifier report details (continued)

Name	Charges by Identifier report
Parameters	• `Date Range` – The list of predefined periods that are used to run the report. • `Start Date` – The lower boundary of the date period that is used to run the report. • `End Date` – The upper boundary of the date period that is used to run the report. • `Rate Table` – Used to specify the rate table. • `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary that denotes the account codes that are displayed in the report. • `Ending Account Code` – The upper boundary that denotes the account codes that are displayed in the report. • `Rate Group` – The rate group to limit the data for. – `All Rate Groups` – Select this option if you want to see charges that are displayed for all rate groups. • `Rate Code` – The Rate Code to limit the data for. – `All Rates` – Select this option if you want to see charges that are displayed for all rates. • `Identifier` – The Identifier to show the charges for in the report. • `Identifier Values` – The Identifier values to show in the report. These values include: – `Top 10` – Select this option to see the identifier values with the highest charges for the selected period. – `Specify` – Select this option to specify the required identifier values. An additional prompt is displayed with the values that can be used. – `All` – Select this option to show all values. **Note:** Some identifiers can contain many values which can take some time to display on the report. • `Graph Type` – `Show Stacked Column Graph` – Select this option if you want to see a Stacked Column Graph. • `Column graph` – `Show Line Graph` – Select this option if you want a line graph displayed. – `Do Not Display Graph` – Select this option if you do not want a graph displayed.

Table 253. Charges by Identifier report details (continued)

Name	Charges by Identifier report
Tables/Views Used	• SCDETAIL – Detail data • CIMSDETAILIDENT – Detail data • CIMSIDENT – Identifier data • SCDIMRATE – Rate data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT – Account contact data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	Depending on the options that are selected in the report properties, the output is displayed in different ways. • If a graph option was selected, the total charges by period or day across all accounts as a stacked column graph or line graph is displayed. If more than 10 identifier values have charges, then only those identifiers with the 10 highest charges for the data period that is specified are displayed. The remaining identifier values are totaled and displayed as the other category. A page is displayed that shows a crosstab that contains the charges for the identifier values by period or day, depending on the option selected. • For each account code, if a graph option was selected, the total charges by period or day across all accounts as a stacked column graph or line graph is displayed. If more than 10 identifier values have charges, then only those identifiers with the 10 highest charges for the data period that is specified are displayed. The remaining identifier values are totaled and displayed as the other category. A page is displayed that shows a crosstab that contains the charges for the identifier values by period or day, depending on the option selected. **Note:** The charges data is reported for normal rate codes. It does not include charges for tiered rates.
Usage	Use the report to monitor the charges for accounts to help you understand any anomalies or get a detailed understanding of the charges.

Detail by Identifier report:

The Detail by Identifier report provides total usage by rate for a selected identifier or range of values for an identifier. Data is displayed for tiered rates at the parent level and non-tiered rates only. This report can be additionally broken down by the resource usage start date using the Date Drilldown parameter as per the Detail by Identifier by Date Report.

Table 254. Detail by Identifier details

Name	Detail by Identifier report
Purpose	Use this report to monitor the total usage by rate for a selected identifier or identifier values selected.
Tivoli Common Reporting folder	Resource Detail
Parameters	• `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Start Value` – The lower boundary of the identifier values to use. • `End Value` – The upper boundary of the identifier values to use. • `Identifier` – The identifier to show the data for. • `Rate Table` – The Rate Table to show the data for (optional). • `Date Drilldown` – Select this to show the resource usage date.
Tables/Views Used	• `SCDETAIL` – Detail data • `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier types • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows the total usage by rate for a selected identifier or range of values for an identifier. This report can be additionally broken down by the resource usage start date using the Date Drilldown parameter.
Usage	The report enables you to monitor the total usage by rate for a selected identifier or range of values for an identifier.

Detail by Multiple Identifiers report:

The Detail by Multiple Identifiers report shows resource units consumed by a maximum of five rate codes and five identifiers. Data is displayed for tiered rates at the parent level and non-tiered rates only. The report can be broken down by the account code using the Show Account Code parameter.

Table 255. Detail by Multiple Identifier details

Name	**Detail by Identifier report**
Purpose	Use this report to monitor the resource units consumed by a maximum of five rate codes and five identifiers.
Tivoli Common Reporting folder	Resource Detail
Parameters	• `Date Range` – The list of predefined periods used to run the report. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – The Rate Table to show the data for (optional). • `Rate Code` – Up to 5 rate codes to display the data for. • `Identifier` – Up to 5 identifiers to display the data for.
Tables/Views Used	• `SCDETAIL` – Detail data • `CIMSDETAILIDENT` – Identifier data • `CIMSIDENT` – Identifier types • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows resource units consumed by a maximum of five rate codes and five identifiers. The report can be broken down by the account code using the Show Account Code parameter.

Table 255. Detail by Multiple Identifier details (continued)

Name	Detail by Identifier report
Usage	The report enables you to monitor the resource units consumed by a maximum of five rate codes and five identifiers.

Usage by Identifier report:

The Usage by Identifier report shows usage by identifier value for a specified identifier type. The available identifiers depend on the data that is loaded. Available identifiers can include, for example, Virtual Image Name, VM Name, and User.

Table 256. Usage by Identifier report details

Name	Usage by Identifier report
Purpose	Use this report to show detailed usage by any provided identifier.
Tivoli Common Reporting folder	Resource Detail

Table 256. Usage by Identifier report details (continued)

Name	Usage by Identifier report
Parameters	• Date Range – The list of predefined periods that are used to run the report. • Start Date – The lower boundary of the date period that is used to run the report. • End Date – The upper boundary of the date period that is used to run the report. • Rate Table – Used to specify the rate table. • Account Structure – The account code structure that is used for the account codes in the reporting. • Account Code Level – The level within the account code structure to display the account codes. • Starting Account Code – The lower boundary that denotes the account codes that are displayed in the report. • Ending Account Code – The upper boundary that denotes the account codes that are displayed in the report. • Rate Group – The rate group to limit the data for. – All Rate Groups – Select this option if you want to see usage displayed for all rate groups. • Rate Code – The Rate Code to limit the data for. • Identifier – The Identifier to show the usage for in the report. • Identifier Values – The Identifier values to show in the report. These values include: – Top 10 – Select this option to see the identifier values with the highest usage for the selected period. – Specify – Select this option to specify the required identifier values. An additional prompt is displayed with the values that can be used. – All – Select this option to show all values. **Note:** Some identifiers can contain many values which can take some time to display on the report. • Graph Type – Show Stacked Column Graph – Select this option if you want to see a Stacked Column Graph. • Column graph – Show Line Graph – Select this option if you want a line graph displayed. – Do Not Display Graph – Select this option if you do not want a graph displayed. • Data Type – Show Period – Select this option to see the usage by period.

Table 256. Usage by Identifier report details (continued)

Name	Usage by Identifier report
Tables/Views Used	• SCDETAIL – Detail data • CIMSDETAILIDENT – Detail data • CIMSIDENT – Identifier data • SCDIMRATE – Rate data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT – Account contact data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	Depending on the options that are selected in the report properties, the output is displayed in different ways. • If a graph option was selected, the total usage by period or day across all accounts as a stacked column graph or line graph is displayed. If more than 10 identifier values have usage, then only those identifiers with the 10 highest usage for the data period that is specified are displayed. The remaining identifier values are totaled and displayed as the other category. A page is also displayed that shows a crosstab that contains the usage for the identifier values by period or day, depending on the option selected. • For each account code, if a graph option was selected, the total usage by period or day across all accounts as a stacked column graph or line graph is displayed. If more than 10 identifier values have usage, then only those identifiers with the 10 highest usage for the data period that is specified are displayed. The remaining identifier values are totaled and displayed as the other category. A page is also displayed that shows a crosstab that contains the usage for the identifier values by period or day, depending on the option selected.
Usage	Use the report to monitor the usage for accounts to help you understand any anomalies or get a detailed understanding of the usage.

Template reports

The **Template reports** folder is used to group all the Cognos Template reports in one location. The folder contains a set of template reports that can be used as a custom report and the report template that can be used by all other reports.

Template report:

The template report contains the header and footer shared by the standard report. See the Rebranding Reports section for further details.

Table 257. Template report details

Name	**Template report**
Purpose	Use this report to amend the shared header and footer used by all of the standard reports.
Tivoli Common Reporting folder	Template
Parameters	• None
Tables/Views Used	• None
Output	This report is not designed to be run.
Usage	The report enables you to rebrand all standard reports in one single action.

Related concepts:

Template reports
Three template reports are provided in the Template folder as part of the SmartCloud Cost Management package. This section describes these reports and how they are used.

Template Account Code Date report:

This report is used as a template for custom reports. As some of the prompt functionality is complex to implement, this report has been created with the functionality for the Account Code and Date prompts.

Table 258. Template Account Code Date report details

Name	**Template Account Code Date report**
Purpose	Use this report to develop custom reports that restrict the data that is displayed by account codes and dates.
Tivoli Common Reporting folder	Template

Table 258. Template Account Code Date report details (continued)

Name	Template Account Code Date report
Parameters	• `Account Structure` – The account code structure used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes at. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used.
Tables/Views Used	• `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	This report is not designed to be run.
Usage	This report is designed to be used as a template for custom reports with the complex prompt functionality pre-written. For more information about this report, see related concept topic in the Administering reports section.

Related concepts:

Template Account Code Date
This report is used as a template for custom reports. As some of the prompt functionality is complex to implement, this report has been created with the basic functionality added.

Template Account Code Year report:

This report is used as a template for custom reports. This report is created with the functionality for the Account Code and Year prompts.

Table 259. Template Account Code Year report details

Name	Template Account Code Year report
Purpose	Use this report to develop custom reports that restrict the data displayed by account codes and year.
Tivoli Common Reporting folder	Template

Table 259. Template Account Code Year report details (continued)

Name	Template Account Code Year report
Parameters	• `Account Structure` – The account code structure used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes at. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Year` – The accounting year that the report is run for.
Tables/Views Used	• `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	This report is not designed to be run
Usage	This report is designed to be used as a template for custom reports with the complex prompt functionality pre-written. For more information about this report, see related concept topic in the Administering reports section.

Related concepts:

Template Account Code Year
This report is used as a template for custom reports. This report is created with the basic functionality added, as the prompt functionality can be complicated to implement.

Top Usage reports

The **Top Usage reports** folder is used to group all the Cognos Top Usage reports in one location. Top Usage reports are used to find accounts and resources with the highest charges.

Top 10 Bar Graph report:

The Top 10 Bar Graph report provides the account codes with the highest charges for the parameters selected as a bar graph and list.

Table 260. Top 10 Bar Graph report details

Name	Top 10 Bar Graph report
Purpose	Use this report to see a list of the top ten accounts with the highest charges.
Tivoli Common Reporting folder	Top Usage Reports

Table 260. Top 10 Bar Graph report details (continued)

Name	Top 10 Bar Graph report
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Rate Table` – Used to specify the rate table.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a Bar Graph of the account codes with the highest charges and a list with the detailed information.
Usage	You can monitor the clients defined in the system to see which are accruing the most charge.

Top 10 Cost report:

The Top 10 Cost report provides the Top N account codes with the highest charges for the parameters selected. For example, if you type 3 as the Top N parameter, the 3 account codes with the highest charges are displayed. If you leave the Top N parameter blank, the account codes with the 10 highest charges are displayed. The report also shows an aggregated total for the other account codes during this period.

Table 261. Top 10 Cost report details

Name	Top 10 Cost report
Purpose	Use this report to view a list of the top N accounts with the highest charges.
Tivoli Common Reporting folder	Top Usage Reports

Table 261. Top 10 Cost report details (continued)

Name	Top 10 Cost report
Parameters	• `Account Structure` – The account code structure used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes at. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Rate Table` – Used to specify the rate table. • `Top N` – The number of account codes to show in the highest charges list. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security sata • `CIMSCALENDAR` – Financial calendar sata
Output	The report shows a list of the account codes with the highest charges. Each account shown can be expanded to show a list of the charges by underlying rate.
Usage	Use this report to monitor the clients defined in the system to see which are accruing the most charge and by which resource.

Top 10 Pie Chart report:

The Top 10 Pie Chart report provides the account codes with the highest charges for the parameters selected as a pie chart and list.

Table 262. Top 10 Pie Chart report details

Name	**Top 10 Pie Chart report**
Purpose	Use this report to see a list of the top ten accounts with the highest charges.
Tivoli Common Reporting folder	Top Usage Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Date Range` – The list of predefined periods to run the report for. • `Start Date` – The lower boundary of the date period used. • `End Date` – The upper boundary of the date period used. • `Rate Table` – Used to specify the rate table.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows a pie chart of the account codes with the highest charges.
Usage	You can monitor the clients defined in the system to see which are accruing the most charge.

Trend reports

The **Trend reports** folder is used to group all the Cognos Trend reports in one location. Trend reports are used to show trends in usage and charges.

Cost Trend report:

The Cost Trend report provides total charges by account code, broken down by rate group and rate code for each period of the year for the parameters selected.

Table 263. Cost Trend report details

Name	**Cost Trend report**
Purpose	Use this report to gain further information about the charges for a particular account.
Tivoli Common Reporting folder	Trend
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Year` – The accounting year that the report is run for. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `CIMSDIMCLIENTCONTACT` – Account contact data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data

Table 263. Cost Trend report details (continued)

Name	Cost Trend report
Output	The report shows charges by Account, Rate Group, and Rate Code. Each account shown can be expanded to show a list of the charges by underlying rate group. Each rate group can be expanded to show a list of charges by rate.
Usage	You can use this report to get detailed information about the charges for an account.

Cost Trend by Rate report:

The Cost Trend by Rate report provides total charges by rate group, broken down by rate and account for each period of the year for the parameters selected.

Table 264. Cost Trend by Rate report details

Name	Cost Trend by Rate report
Purpose	Use this report to gain further information about the charges for a particular rate group and how they are used for each rate and by who.
Tivoli Common Reporting folder	Trend
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Year` – The accounting year that the report is run for. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.

Table 264. Cost Trend by Rate report details (continued)

Name	Cost Trend by Rate report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT– Account contact data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows charges by rate group, rate, and account. Each rate group shown can be expanded to show a list of the charges by underlying rate. Each rate can be expanded to show a list of charges by account.
Usage	Use this report to get detailed information about the charges for a rate group.

Cost Trend Graph report:

The Cost Trend Graph report provides total charges for all account codes by period, and for each account by period.

Table 265. Cost Trend Graph report details

Name	Cost Trend Graph report
Purpose	Use this report to gain further information about the charges across all accounts and within individual accounts.
Tivoli Common Reporting folder	Trend
Parameters	• Account Structure – The account code structure that is used for the account codes in the reporting. • Account Code Level – The level within the account code structure to display the account codes. • Starting Account Code – The lower boundary denoting the account codes displayed in the report. • Ending Account Code – The upper boundary denoting the account codes displayed in the report. • Rate Table – Used to specify the rate table. • Rate Pattern – Used to specify the rate pattern type. • Year – The accounting year that the report is run for.

Table 265. Cost Trend Graph report details (continued)

Name	Cost Trend Graph report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSDIMCLIENT – Account data • CIMSDIMCLIENTCONTACT– Account contact data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data. • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows charges for all accounts as a bar chart with the charge broken down by period. A table is shown with the totals by period. The report also shows charges by account, broken down by period as a chart and a table.
Usage	You can use this report to get further information about the charges for all and specific accounts.

Resource Usage Trend report:

The Resource Usage Trend report provides total resource usage by rate code for each month of the year for the parameters selected. It is ordered by account code, rate group, and rate code.

Table 266. Resource Usage Trend details

Name	Resource Usage Trend report
Purpose	The Resource Usage Trend report provides total resource usage by rate code for each month of the year for the parameters selected.
Tivoli Common Reporting folder	Trend

Table 266. Resource Usage Trend details (continued)

Name	Resource Usage Trend report
Parameters	• `Account Code Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Rate Group` – The rate group used to show the rate details (optional). • `Year` – The accounting year that the report is run for. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report provides total resource usage by rate code for each month of the year for the parameters selected. It is ordered by account code, rate group, and rate code.
Usage	The report provides total resource usage by rate code for each month of the year for the parameters selected.

Usage Trend Graph report:

The Usage Trend Graph report is used to show the total usage for all rate codes by period.

Table 267. Usage Trend Graph report details

Name	Usage Trend Graph report
Purpose	Use this report to gain further information about the Usage across all rates.
Tivoli Common Reporting folder	Trend
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Rate Group` – The rate group used to show the rate details (optional). • `Year` – The accounting year that the report is run for. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSDIMCLIENT` – Account data • `SCDIMRATE` – Rate data. • `CIMSCONFIGACCOUNTLEVEL` – Account structure data. • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data
Output	The report shows usage for all rates as a bar chart with the usage broken down by period. A table is also shown with the totals by period.
Usage	Use this report to get further information about the usage for rate codes.

Variance reports

The **Variance reports** folder is used to group all the Cognos Variance reports. Variance reports are period comparison reports for usage and charges.

Cost Variance report:

The Cost Variance report provides a comparison of charges by account code, rate code description, and rate group for a specified period and a previous period, for the parameters selected.

Table 268. Cost Variance report details

Name	Cost Variance report
Purpose	Use this report to compare the charges for a period with the previous period or the same period last year.
Tivoli Common Reporting folder	Variance Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Table` – Used to specify the rate table. • `Rate Pattern` – Used to specify the rate pattern type. • `Year` – The accounting year that the report is run for. • `Period` – The accounting period that the report is run for. • `Compare` – The corresponding period to compare the chosen period with. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.
Tables/Views Used	• `SCSUMMARY` – Summary data • `CIMSCLIENT` – Account data • `SCDIMRATE` – Rate data • `CIMSCONFIGACCOUNTLEVEL` – Account structure data • `CIMSCONFIG` – Configuration data • `CIMSDIMUSERRESTRICT` – User security data • `CIMSCALENDAR` – Financial calendar data

Table 268. Cost Variance report details (continued)

Name	Cost Variance report
Output	The report shows charges by Account, Rate Group, and Rate Code for the chosen period and previous period. The report also shows the variance and percentage variance between the 2 periods.
Usage	You can use this report to understand how charges are accruing over time.

Resource Variance report:

The Resource Variance report provides a comparison of usage by account code, rate code description, and rate group for a specified period and a previous period, for the parameters selected.

Table 269. Resource Variance report details

Name	Resource Variance report
Purpose	Use this report to compare the usage for a period with the previous period or the same period last year.
Tivoli Common Reporting folder	Variance Reports
Parameters	• `Account Structure` – The account code structure that is used for the account codes in the reporting. • `Account Code Level` – The level within the account code structure that is used to display the account codes. • `Starting Account Code` – The lower boundary denoting the account codes displayed in the report. • `Ending Account Code` – The upper boundary denoting the account codes displayed in the report. • `Rate Pattern` – Used to specify the rate pattern type. • `Year` – The accounting year that the report is run for. • `Period` – The accounting period that the report is run for. • `Compare` – The corresponding period to compare the chosen period with. • `Show Consolidated` – Select this option if you want to display non-tiered rates and tiered rates at the parent level. • `Show Child Rate` – Select this option if you want to display non-tiered rates and tiered rates at the child level.

Table 269. Resource Variance report details (continued)

Name	Resource Variance report
Tables/Views Used	• SCSUMMARY – Summary data • CIMSCLIENT – Account data • SCDIMRATE – Rate data • CIMSCONFIGACCOUNTLEVEL – Account structure data • CIMSCONFIG – Configuration data • CIMSDIMUSERRESTRICT – User security data • CIMSCALENDAR – Financial calendar data
Output	The report shows usage by Account, Rate Group, and Rate Code for the chosen period and previous period. The report also shows the variance and percentage variance between the 2 periods.
Usage	You can use this report to understand resource usage over time.

Tables

This section describes the tables that are available in the product.

CIMSAccountCodeConversion Table

The CIMSAccountCodeConversion table stores conversion definitions associated with process definitions.

Field Name	Key	Type	Field Description
AccountCodeConversionID	PK	bigint	The unique identifier for the Conversion definition.
ProcDefinition		varchar(100)	The Process Definition this Conversion is associated with.
Description		varchar(255)	A description of this Conversion definition
SourceAccountCodeIdentValueLow		varchar(128)	The source Account Code or Identifer value or Low Identifier value.
SourceAccountCodeIdentValueHigh		varchar(128)	The source High Identifier value.
EffectiveDate		timestamp	The date this Conversion becomes effective.
ExpiryDate		timestamp	The date this Conversion expires.
TargetAcctCodeIdentValue		varchar(128)	The target Account Code or Identifier value.

a. PK = Primary Key

CIMSCalendar Table

The CIMSCalendar table defines the billing periods for a billing organization.

Field Name	Key	Type	Field Description
Year	PK[a]	int	The calendar year.
Period	PK[a]	int	The calendar period (1 - 13).
PeriodBeginDate		datetime	The beginning date of the period
PeriodEndDate		datetime	The ending date of the period
CloseDate		datetime	The processing close date.

a. PK = Primary Key

CIMSClient Table

The CIMSClient table is the primary table for storing client information.

All other client tables are related to this table.

Field Name	Key	Type	Field Description
AccountCode	PK[a]	char(127)	The account code for the client.
AltAcctCode		char(127)	The alternate account code for the client (optional).
AccountName		varchar(255)	The client name.
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.
RateTable		char(30)	The rate table for the client.
InvoiceContact		int	The index number for the client contact name that is displayed on the invoice.
ActionCode1 - ActionCode8		char(1)	These user-defined codes are strictly for reporting purposes and can be used in the custom reporting process for selecting data.

a. PK = Primary Key

CIMSClientBudget Table

The CIMSClientBudget table stores budget and actual information for amounts and resource units, by account code, date, and rate code.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.

Field Name	Key	Type	Field Description
AccountCode	PK[a]	char(127)	The account code for the client.
RateCode	PK[a]	char(30)	A valid rate code for an individual resource budget or TOTAL for an overall account budget.
Year	PK[a]	int	The budget year.
Period	PK[a]	int	The budget period (0 - 13).
BudgetAmt		decimal(19,4)	The budget monetary amount for the period.
ActualAmt		decimal(19,4)	Deprecated. The actual amount is in the CIMSSummary table.
BudgetUnits		decimal(18,5)	The budget units for the period.
ActualUnits		decimal(18,5)	Deprecated. The actual units consumed are in the CIMSSummary table.

a. PK = Primary Key

CIMSClientContact Table

The CIMSClientContact table stores information about client contacts.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.
AccountCode	PK[a]	char(127)	The account code for the client.
ContactIdx	PK[a]	int	The index number for the contact.
AddressLine1		varchar(255)	Address Line 1
AddressLine2		varchar(255)	Address Line 2
AddressLine3		varchar(255)	Address Line 3
AddressLine4		varchar(255)	Address Line 4
AddressLine5		varchar(255)	Address Line 5
ContactName		varchar(255)	The name of the client contact.
Department		varchar(255)	The contact department.
Office		varchar(255)	The contact office.
Comments		varchar(255)	Comments related to the contact.

a. PK = Primary Key

CIMSClientContactNumber Table

The CIMSClientContactNumber table stores numbers (phone, fax, etc.) and e-mail addresses for contacts.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.
AccountCode	PK[a]	char(127)	The account code for the client.
ContactIdx	PK[a]	int	The index number for the contact.
NumberIdx	PK[a]	int	The index number for the contact number or e-mail address.
NumberDescription		varchar(255)	The type of number (VOICE, FAX, E-MAIL, etc.)
NumberValue		varchar(255)	The contact number or e-mail address.

a. PK = Primary Key

CIMSConfig Table

The CIMSConfig table defines configuration options that have a GUI option in Administration Console.

One CIMSConfig record is allowed per SmartCloud Cost Management database.

Field Name	Key	Type	Field Description
OrgName		varchar(255)	The organization name
AddressLine1		varchar(255)	Address Line 1
AddressLine2		varchar(255)	Address Line 2
AddressLine3		varchar(255)	Address Line 3
AddressLine4		varchar(255)	Address Line 4
InvoiceNo		int	Deprecated. This is stored in the CIMSConfigOptions table.
InvoiceNumberType		char(1)	Deprecated.
Use13Periods		char(1)	Indicates whether 13 periods are used: • Y (Yes) • N (No)
UseCalender		char(1)	Indicates whether the SmartCloud Cost Management calendar is used: • Y (Yes) • N (No)
DatabaseVersion		char(7)	The SmartCloud Cost Management database version number.
LastProcessUID		int	Used by CIMSAcct and CIMSBill.

Field Name	Key	Type	Field Description
AccountParameterSelectionLevel		int	In SmartCloud Cost Management reporting, this setting determines the level of account codes that are displayed in the **Starting Account Code** and **Ending Account Code** parameter lists.
DBConnectionTimeout		int	The maximum number of seconds that a database may respond to a query before timing out
LastReportingDate		datetime	The last reporting date for reports that users who do not have administrative privileges can view.

CIMSConfigAccountLevel Table

The CIMSConfigAccountLevel table defines the account code structure.

Field Name	Key	Type	Field Description
AccountStructureName		char(32)	The account code structure name (e.g., Standard).
StartPosition		int	The starting offset position for the account code.
AccountLevel		int	The account code level.
AccountLength		int	The length of the account code level.
AccountDescription		varchar(255)	A description of the account code level.

CIMSConfigOptions Table

The CIMSConfigOptions table defines configuration options that the administrator can add manually in Administration Console.

Field Name	Key	Type	Field Description
PropertyName	PK[a]	varchar(80)	The name of the configuration option.
PropertyValue		varchar(255)	The value for the configuration option.
PropertyType		int	The data type of the configuration option: • 1 (String) • 2 (Integer) • 3 (Floating Point) • 4 (Date/Time)
Sequence		int	For future use.
PropertyDescription		varchar(255)	A description of the configuration option.

a. PK = Primary Key

CIMSCPUNormalization Table

The CIMSCPUNormalization table allows for differences in CPU speed.

Field Name	Key	Type	Field Description
SystemID	PK[a]	char(32)	The system identifier for the computer that you want to normalize. For z/OS, this is the four-character System Model ID. For UNIX and Windows, it is the computer name.
WorkID	PK[a]	char(32)	The subystem name (e.g., CICS region name, DB2 plan name, Oracle instance, etc.).
Factor		decimal(10,4)	The normalization factor.

a. PK = Primary Key

CIMSDetailIdent Table

The CIMSDetailIdent table stores the information contained in the Ident file. The Ident file is produced by the Bill program.

The table contains the identifier values collected from the records in the input CSR or CSR+ file. The identifier names are stored in the CIMSIdent table and are represented by a number assigned to the identifier name.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	The load tracking unique identifier from the CIMSLoadTracking table.
DetailUID	PK[a]	int	The unique identifier that tracks the Detail load.
DetailLine	PK[a]	int	The number of the record in the input CSR or CSR+ file that contains the identifier (i.e., 1 if it is the first record, 2 if it is the second record, etc.).
IdentNumber	PK[a]	int	A sequential number assigned to the identifier in the CIMSIdent table.
IdentValue		varchar(255)	The value of the identifier.

a. PK = Primary Key

CIMSDIMClient Table

The CIMSDimClient table consolidates client information and the users who can access them for use with Tivoli Common Reporting Cognos reports..

Field Name	Key	Type	Field Description
LDAPUserUID	PK[a]	varchar(255)	The Central Registry user ID.
UserID		char(100)	The SmartCloud Cost Management user ID.
GroupID		char(100)	Deprecated.
AccountCode	PK[a]	char(127)	The client account code that can be viewed. If the group has access to all clients, this field is blank.
AltAcctCode		char(127)	The alternate account code for the client (optional).
AccountName		varchar(255)	The client name.
LoadTrackingUID		int	The unique identifier that tracks the rate when loaded to the database from a location outside of SmartCloud Cost Management.
RateTable		char(30)	The name of the rate table that contains the rate code.
InvoiceContact		int	The index number for the client contact name that is displayed on the invoice.
ActionCode1 - ActionCode8		char(1)	These user-defined codes are strictly for reporting purposes and can be used in the custom reporting process for selecting data.

a. PK = Primary Key

CIMSDIMClientContact Table

The CIMSDIMClientContact table consolidates the client contact information and the users who can access them for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
LDAPUserID	PK[a]	varchar(255)	The Central Registry user ID.
UserID		char(100)	The SmartCloud Cost Management user ID.
GroupID		char(100)	Deprecated.
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.
AccountCode	PK[a]	char(127)	The account code for the client.
ContactIdx	PK[a]	int	The index number for the contact.

Field Name	Key	Type	Field Description
AddressLine1		varchar(255)	Address Line 1
AddressLine2		varchar(255)	Address Line 2
AddressLine3		varchar(255)	Address Line 3
AddressLine4		varchar(255)	Address Line 4
AddressLine5		varchar(255)	Address Line 5
ContactName		varchar(255)	The name of the client contact.
Department		varchar(255)	The contact department.
Office		varchar(255)	The contact office.
Comments		varchar(255)	Comments related to the contact.

a. PK = Primary Key

CIMSDIMClientContactNumber Table

The CIMSDIMClientContactNumber table consolidates the information about client contact numbers and the users who can access them for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
LDAPUserID	PK[a]	varchar(255)	The Central Registry user ID.
UserID		char(100)	The Usage and Accounting Manager user ID.
GroupID		char(100)	Deprecated.
LoadTrackingUID		int	A unique identifier that tracks the client when loaded to the database from a location outside of SmartCloud Cost Management.
AccountCode	PK[a]	char(128)	The account code for the client.
ContactIdx	PK[a]	int	The index number for the contact.
NumberIdx	PK[a]	int	The index number for the contact number or email address.
NumberDescription		varchar(255)	The type of number (VOICE, FAX, E-MAIL, and so on.)
NumberValue		varchar(255)	The contact number or email address.

a. PK = Primary Key

CIMSDIMUserRestrict Table

The CIMSDimUserRestrict table consolidates the relevant information about users and their SmartCloud Cost Management data access privileges for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
LDAPUserID		varchar(255)	The Central Registry User ID.
UserID	PKa	char(100)	The SmartCloud Cost Management user ID.
GroupID		char(100)	Deprecated.
LastReportingDate		datetime	Indicates the Last Reporting Date for reports that users in this group who do not have administrative privileges can view (if set).
AccountCode	PK[a]	char(127)	The client account code that can be viewed. If the group has access to all clients, this field is blank.
AccountLength		int	The length of the account code level.
LastSortedSeqChar		varchar(4000)	A string consisting of the Last Sorted Sequence Character repeated for use in report filters.

a. PK = Primary Key

CIMSIdent Table

The CIMSIdent table stores all identifier names contained in the input CSR or CSR+ files processed.

As a file is processed, the identifier names in the file records are added to this table. This table will continue to store identifier names even if the identifiers are no longer used by records loaded in the database. To delete unused identifier names if needed, go to the Administration Console Identifier List Maintenance page.

Field Name	Key	Type	Field Description
IdentNumber	PK[a]	int	A sequential number assigned to the identifier.
IdentName		varchar(32)	A short name for the identifier.
IdentDescription		varchar(255)	A full description of the identifier name (optional).
IdentActiveFlag		char(1)	Indicates whether Active or Inactive has been selected for custom reports: • Blank (Active) • I (Inactive)
IdentReportFlag		char(1)	User-defined report flag.

a. PK = Primary Key

CIMSLoadTracking Table

The CIMSLoadTracking table records the information about data that has been loaded into the database.

This enables database loads to be deleted if required.

Field Name	Key	Type	Field Description
LoadTrackingUID	PK[a]	int	The unique identifier for the database load.
FileType		char(8)	The type of file processed: • Summary • Detail (CIMSBill Detail) • Ident • Resource (CIMSAcct Detail)
FileName		varchar(255)	The path and name of the file that was processed
DateLoaded		datetime	The date and time the input CSR or CSR+ file was processed by CIMSAcct.
DateDeleted		datetime	The date and time the load was removed from the database.
TotalRecsLoaded		int	The number of records loaded.
StartDate		datetime	The accounting start date for Summary, Detail, and Ident records. The usage start date for resource records.
EndDate		datetime	The accounting end date for Summary, Detail, and Ident records. The usage end date for resource records.
SourceSystem		varchar(32)	The name of the computer that produced the data.
SourceFeed		varchar(32)	The system that produced the data (e.g., IIS, Exchange, z/OS, etc.)
DateArchived		datetime	The date and time that the load was archived.
ArchiveLocation		varchar(255)	The location of the archived load.
LoadGroupUID		int	The unique identifier for the group that the database load belongs to.

a. PK = Primary Key

CIMSProcDefinitionMapping Table

The CIMSProcDefinitionMapping table defines the conversion and proration mapping types associated with process definitions.

Field Name	Key	Type	Field Description
ProcDefinitionMappingID	PK	bigint	The unique identifier for the process definition mapping.
ProcDefinition		varchar(100)	The process definition this mapping is associated with.

Field Name	Key	Type	Field Description
MappingObject		varchar(10)	Specifies the mapping object, *CONVERSION* or *PRORATION*.
MappingDirection		varchar(10)	Specifies the mapping direction, *SOURCE* or *TARGET*.
MappingType		varchar(20)	Specifies the mapping type, *IDENTIFIER* or *ACCOUNTCODE*.
IdentType		varchar(255)	Unused
AccountStructureName		char(32)	Unused
AccountLockLevels		int	Unused
ValidateTo		int	Unused
HighlowMapping		smallint	Specifies if a conversion mapping uses Low and High Identifier values.

a. PK = Primary Key

CIMSProcessDefinition Table

The CIMSProcessDefinition table defines process definitions.

Field Name	Key	Type	Field Description
ProcDefinition	PK	varchar(100)	The name of the process definition.
LockDate		timestamp	The **Lockdown date** of the process definition. No changes can be made to the conversion/proration mappings prior to this date.
UseCalendar		smallint	Unused
ConversionUpdates		int	Specifies how the Process Definition was created. If set to 0, the Process Definition was created manually using the Business Rules panel. If set to 1, the Process Definition was created/updated by the UpdateConversionFromRecord stage.

a. PK = Primary Key

CIMSProrateSource Table

The CIMSProrateSource table stores proration definitions associated with process definitions.

Field Name	Key	Type	Field Description
ProrateSourceID	PK	bigint	The unique identifier for the proration definition.
ProcDefinition		varchar(100)	The process definition this proration is associated with.

Field Name	Key	Type	Field Description
SourceAccountCodeIdentValue		varchar(128)	The source Account Code or Identifer value.
EffectiveDate		datetime	The date this proration becomes effective.
ExpiryDate		datetime	The date this proration expires.

a. PK = Primary Key

CIMSProrateTarget Table

The CIMSProrateTarget table stores target proration details.

Field Name	Key	Type	Field Description
ProrateSourceID	PK	bigint	The identifier for the proration source.
TargetAcctCodeIdentValue	PK	varchar(128)	The target Account Code or Identifier value.
RateCode	PK	char(30)	The Rate Code used in the proration target.
ProratePercent		decimal (6,10)	The percentage at which the target proration is defined.

a. PK = Primary Key

CIMSRateGroup Table

The CIMSRateGroup table defines the rate groups.

Field Name	Key	Type	Field Description
RateGroup	PK[a]	int	A sequential number assigned to the rate group.
ParentUID		int	For future use.
GroupTitle		char(300)	The rate group title.
GroupTitleLong		varchar(300)	A description of the rate group.
PrintIndex		int	The order in which the rate group is displayed in reports.
OfferingCode		char(30)	The code used to identify the offering.

a. PK = Primary Key

CIMSRateIdentifiers Table

The CIMSRateIdentifiers table correlates rate codes to identifiers to enable drill down on the units consumed for a resource by identifier.

Field Name	Key	Type	Field Description
RateCode	PK[a]	char(30)	The rate code.
IdentNumber	PK[a]	int	A sequential number assigned to the identifier for the rate code.

a. PK = Primary Key

CIMSReportCustomFields Table

The CIMSReportCustomFields table stores the values for the user-defined reports that can be stored in the database and supplied to the report automatically at run time.

Field Name	Key	Type	Field Description
ReportID	PK[a]	int	The unique identifier for the report.
RateCode		char(30)	The rate code
ReportPosition	PK[a]	int	The position within the report where the values are used beginning with 1 and continuing sequentially.
DecimalPlaces		int	The number of decimal places to display for the resource usage amount in reports.

a. PK = Primary Key

CIMSReportDistribution Table

The CIMSReportDistribution table store the values for the report distribution requests.

These requests are associated with a report cycle.

Field Name	Key	Type	Field Description
UserID		char(100)	The SmartCloud Cost Management user ID for the user assigned to the report.
GroupID		char(100)	The SmartCloud Cost Management user group ID for the user group assigned to the report.
ReportID		int	The unique identifier for the report.
ReportCycleID	PK[a]	int	The unique identifier for the report cycle.
RequestID	PK[a]	int	The unique identifier for the report distribution request.

Field Name	Key	Type	Field Description
ReportFormat		int	The unique identifier for the report format.

a. PK = Primary Key

CIMSReportDistributionParm Table

The CIMSReportDistributionParm table stores the values for each report assigned to a report cycle.

Field Name	Key	Type	Field Description
ReportID	PK[a]	int	The unique identifier for the report.
ReportCycleID	PK[a]	int	The unique identifier for the report cycle.
RequestID	PK[a]	int	The unique identifier for the report distribution request.
ParmName	PK[a]	varchar(255)	The parameter name.
ParmValue		varchar(255)	The parameter value.

a. PK = Primary Key

CIMSReportDistributionType Table

The CIMSReportDistributionType table stores the values for report formats.

Field Name	Key	Type	Field Description
TypeID	PK[a]	int	The unique identifier for the report format.
TypeValue		varchar(255)	The description of the report format.

a. PK = Primary Key

CIMSReportGroup Table

The CIMSReportGroup table stores information about report groups.

Field Name	Key	Type	Field Description
ReportGroupID	PK[a]	int	The unique identifier for the report group
ReportGroupName		varchar(128)	The report group name
ReportGroupDesc		varchar(255)	A description of the report group.

a. PK = Primary Key

CIMSReportToReportGroup Table

The CIMSReportToReportGroup table relates reports to report groups.

Field Name	Key	Type	Field Description
MemberID	PK[a]	int	The unique identifier of the report.
MemberType	PK[a]	char(1)	• R (Report) • G (Graph)
ReportGroupId	PK[a]	int	The unique identifier for the report group.
Sequence		int	The order of the report in the report group list.

a. PK = Primary Key

CIMSSummaryToDetail Table

The CIMSSummaryToDetail table is obsolete.

Field Name	Key	Type	Field Description
SummaryLoadTrackingUID	PK[a]	int	The load tracking unique identifier from the CIMSLoadTracking table.
DetailUID	PK[a]	int	The unique identifier that tracks the Detail load.
StartDate		datetime	The accounting start date.
EndDate		datetime	The accounting end date.
DetailConverted		char(1)	Indicates whether the Detail records have been converted: Y (Yes) Null (No)
SummaryConverted		char(1)	Indicates whether the Summary records have been converted: Y (Yes) Null (No)

a. PK = Primary Key

CIMSTransaction Table

The CIMSTransaction table stores miscellaneous, recurring, and credit transactions.

Field Name	Key	Type	Field Description
TransactionUID	PK[a]	char(32)	The unique identifier for the transaction.
AccountCode		char(127)	The account code for the transaction.

Field Name	Key	Type	Field Description
TransactionType		char(1)	The transaction type: • M (Miscellaneous) • R (Recurring) • C (Credit)
ShiftCode		char(1)	The shift code for the transaction.
RateCode		char(30)	The rate code for the transaction.
ResourceAmount		decimal(18,5)	The amount of the transaction.
Frequency1		int	Applicable only to recurring transactions. The frequency that the transaction should occur (every month, every 6 months, etc.). Frequency is based on the calendar year (January-December) • 1 (monthly) • 2 (every other month) • 3 (every quarter) • 4 (every four months) • 6 (every six months) • 12 (once a year)
Frequency2		int	Applicable only to recurring transactions. The period in which the transaction should be processed. This value corresponds to the value in the Frequency1 field. For example, if the value in the Frequency1 field is 6, a value of 1 in this field indicates the first month of a 6 month period (January or July).
FromDate/ToDate		datetime	Applicable only to miscellaneous and credit transactions. The date range that the transaction occurred.
DateTimeSent		datetime	The date and time that the transaction was exported to a flat file.
DateTimeModified		datetime	The date and time that the transaction was last modified.
DateTimeEntered		datetime	The date and time that the transaction was created.
DateTimeStartProcessing		datetime	Applicable only to recurring transactions. The first day that the transaction will be processed.
DateTimeStopProcessing		datetime	Applicable only to recurring transactions. The last day that the transaction will be processed.
UserID		varchar(255)	The SmartCloud Cost Management user ID of the person who entered the transaction.

Field Name	Key	Type	Field Description
DateTimeDeleted		datetime	The date and time that the transaction was deleted.
Note		varchar(255)	A description of the transaction.

a. PK = Primary Key

CIMSUser Table

The CIMSUser table stores the user options entered in the Administration Console User Maintenance page.

Field Name	Key	Type	Field Description
UserID	PKa	char(100)	The SmartCloud Cost Management user ID.
PasswordHash		varchar(255)	A hash of the user password.
SessionID		int	A unique session identifier.
UserFullName		varchar(255)	The full name of the user.
DomainUserName		varchar(255)	The Windows domain and user name for the user.
EmailToAddress		varchar(255)	The e-mail address for the user.
ReportOption		char(1)	This field indicates if a user is using Tivoli Common Reporting (T), or none.
LDAPUserID		varchar(255)	The Central Registry User ID.
LDAPUserUID		varchar(255)	The full version of the Central Registry User ID.

a. PK = Primary Key

CIMSUserConfigOptions Table

The CIMSUserConfigOptions table stores the user configuration options entered in the Administration Console User Configuration Option Maintenance page.

Field Name	Key	Type	Field Description
PropertyName	PK[a]	varchar(80)	The name of the configuration option.
UserID	PK[a]	char(100)	The SmartCloud Cost Management user ID.
PropertyValue		varchar(255)	The value for the configuration option.
PropertyType		int	The data type of the configuration option: • 1 (String) • 2 (Integer) • 3 (Floating Point) • 4 (Date/Time)
Sequence		int	For future use.

Field Name	Key	Type	Field Description
PropertyDescription		varchar(255)	A description of the configuration option.

a. PK = Primary Key

CIMSUserGroupAccountCode Table

The CIMSUserGroupAccountCode tables defines the account codes that users within a user group can view.

Field Name	Key	Type	Field Description
GroupID	PK[a]	char(100)	The user group ID.
AccountCode	PK[a]	char(127)	The client account code that can be viewed. If the group has access to all clients, this field is blank.

a. PK = Primary Key

CIMSUserGroupAccountStructure Table

The CIMSUserGroupAccountStructure table stores the account code structures assigned to a user group.

Field Name	Key	Type	Field Description
GroupID	PK[a]	char(100)	The user group ID.
AccountStructureName	PK[a]	char(32)	The name of the account code structure.
GroupDefault		int	Indicates whether the account code structure is the default: 1 (default) 0 (not default)

CIMSUserGroupConfigOptions Table

The CIMSUserGroupConfigOptions table stores the user configuration options entered in the Administration Console User Group Configuration Option Maintenance page.

Field Name	Key	Type	Field Description
PropertyName	PK[a]	varchar(80)	The name of the configuration option.
GroupID	PK[a]	char(100)	The SmartCloud Cost Management user group ID.
PropertyValue		varchar(255)	The value for the configuration option.

Field Name	Key	Type	Field Description
PropertyType		int	The data type of the configuration option: • 1 (String) • 2 (Integer) • 3 (Floating Point) • 4 (Date/Time)
Sequence		int	For future use.
PropertyDescription		varchar(255)	A description of the configuration option.

a. PK = Primary Key

CIMSUserGroupReport Table

The CIMSUserGroupReport tables defines the reports that users within a user group can view.

If a group has access to all reports, it is not included in this table.

Field Name	Key	Type	Field Description
GroupID	PK[a]	char(100)	The SmartCloud Cost Management user group ID.
ReportID	PK[a]	int	The unique identifier of the report.
ReportExecute		char(1)	Permission for users in this group to run the report: • Y (Yes) • N (No)

a. PK = Primary Key

CIMSUserGroup Table

The CIMSUserGroup table relates users to user groups

.

Field Name	Key	Type	Field Description
GroupID	PK[a]	char(100)	The user group ID.
GroupFullName		varchar(255)	The full name of the group.
TransactionSecurity		char(1)	Indicates whether group has transaction maintenance privilege: 0 (No) 9 (Yes)
RateSecurity		char(1)	For future use.
ClientSecurity		char(1)	For future use.

Field Name	Key	Type	Field Description
GroupType		char(1)	Type of user: 1 (Client) 9 (Admin)
AllowFinModeler		char(1)	Indicates whether group has access to SmartCloud Cost Management Financial Modeler: 0 (No access) 9 (Access) **Note:** For IBM Cloud Orchestrator, this field is not applicable.
AllowWebConsole		char(1)	This field is deprecated.
LastReportingDate		datetime	Indicates the Last Reporting Date for reports that users in this group who do not have administrative privileges can view (if set).
UserID		char(100)	The user ID.
GroupID		char(100)	The group ID.

a. PK = Primary Key

CIMSUserGroupProcessDef Table

The CIMSUserGroupProcessDef tables stores the process definitions assigned to a user group.

Field Name	Key	Type	Field Description
GroupID	PK	char(100)	The user group ID.
ProcDefinition	PK	varchar(100)	The process definition that can be viewed/edited. If the group has access to all process definitions, this field is blank.

a. PK = Primary Key

CIMSUsertoUserGroup Table

The CIMSUsertoUserGroup table links the user to the usergroup.

Field Name	Key	Type	Field Description
UserID		char(100)	The user ID.
GroupID		char(100)	The group ID.

a. PK = Primary Key

SCDetail Table

The SCDetail table stores the information that is contained in the Detail file. The Detail file is produced by the Bill program.

The table contains the resource information collected from the records in the input CSR or CSR+ file. Each record in this table represents a single resource used/time period. This is the most granular form of the resource data stored in the database.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	The load tracking unique identifier from the CIMSLoadTracking table.
DetailUID		int	The unique identifier that tracks the Detail load.
DetailLine		int	The number of the record in the input CSR or CSR+ file that contains the resource (i.e., 1 if it is the first record, 2 if it is the second record, etc.).
AccountCode		char(127)	The account code.
Aggregate		int	A count of the number of original input records that have been aggregated into this record.
StartDate		datetime	The resource usage start date.
EndDate		datetime	The resource usage end date.
ShiftCode		char(1)	The resource usage shift code.
AuditCode		varchar(255)	The audit code (if applicable).
SourceSystem		char(8)	The system identifier for this resource.
RateCode		char(30)	The rate code of this resource.
ResourceUnits		decimal(18,5)	The quantity of the resource used.
AccountingStartDate		datetime	The accounting start date.
AccountingEndDate		datetime	The accounting end date.
UnitCode		int	The units the rate code is measured in.
RateCodeEffDate		datetime	The effective date for the rate code.
RateTable		char(30)	The name of the rate table that contains the rate code.
MoneyValue		decimal(19,6)	The total charge for the resource units after the rate value is applied.
RateValue		decimal(18,8)	The resources rate value.

SCDIMCalendar Table

The SCDIMCalendar table consolidates the information about the financial calendar for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
Curr_Date	PK[a]	datetime	Date
Curr _Date_StartDate		datetime	Date with Start Time for Day
Curr _Date_EndDate		datetime	Date with End Time for Day
Curr _Week		int	Week Number of Date
Curr _Week_StartDate		datetime	Start Date of Week
Curr _Week_EndDate		datetime	End Date of Week
Curr _Period		int	Financial Period of Date
Curr _Period_StartDate		datetime	Start Date of Financial Period
Curr _Period_EndDate		datetime	End Date of Financial Period
Curr _Year		int	Year of Date
Curr _Year_StartDate		datetime	Start Date of Year
Curr _Year_EndDate		datetime	End Date of Year

a. PK = Primary Key

SCDIMRate Table

The SCDimRate table consolidates the information about rate groups and rates for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
RateGroup	PK[a]	int	A sequential number assigned to the rate group.
ParentUID		int	For future use.
GroupTitle		varchar(300)	The rate group title.
GroupTitleLong		varchar(300)	A description of the rate group.
PrintIndex		int	The order in which the rate group is displayed in reports.
ParentLoadTrackingUID		int	The unique identifier that tracks the parent rate when loaded to the database from a location outside of SmartCloud Cost Management.
ParentRateCode		char(30)	The code for the parent rate.
ParentRateIndex		decimal(10,4)	The index number for the parent rate code.
ParentRateValue		decimal(18,8)	The per unit value for the parent rate code.
ParentResourceValue		decimal(19,4)	Deprecated
ParentDiscountDollars		decimal(10,4)	Deprecated
ParentDiscount		decimal(7,4)	Deprecated
ParentDiscountIndex		int	Deprecated
ParentDescription		varchar(255)	parent description

Field Name	Key	Type	Field Description
ParentFactor		decimal(15,8)	A user-defined resource conversion factor value for the resource unit value in reports used by the parent rate code.
ParentRateValue1		char(1)	Indicates whether 4 decimal positions are used in the rate value in reports for the parent rate code: • F (4 digits are used) • Blank (4 digits are used [the default])
ParentRateValue2		char(1)	Indicates whether the rate is per resource unit or per 1000 units in reports for the parent rate code: • M (Per 1000 units) • Blank (Per unit)
ParentRateValue3		char(1)	Indicates the resource conversion factor for the resource unit value in reports for the parent rate code. • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.) • Blank (No conversion factor)
ParentRateValue4		char(1)	Indicates whether the parent rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations)
ParentRateValue5		char(1)	Indicates the number of decimal digits that appear in the resource units value in reports for the parent rate code.: • 0-5 (0 through 5 digits) • Blank (2 digits, the default)
ParentRateValue6		char(1)	For future use.

Field Name	Key	Type	Field Description
ParentRateValue7		char(1)	Indicates whether the resource units for the rate code are considered a monetary amount rather than units if utilization: • $ (flat fee) • Blank (not flat fee)
ParentRateValue8		char(1)	User-defined report flag.
ParentRateValue9		char(1)	For future use.
ParentRateValue10		char(1)	User-defined report flag.
ParentRateValue11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y– yes • N –no
ParentComments		varchar(255)	Comments for the parent rate code.
ParentDetailDescription		varchar(255)	User-defined description of the rate code for custom reports.
ParentCurrencySymbol		varchar(3)	User-defined currency symbol for custom reports used by the parent rate code.
ParentRateTypeCode		int	This is the type of parent rate code used for display purposes.
ParentTierMethod		int	The tiering method for the parent rate. This can have the following values: • 1 – Non-Tiered • 2 – Individual • 3 – Highest • 4 – Individual Percent • 5 – Highest Percent
ParentRateTypeCode		int	This is the type of rate code used for display purposes. This can have the following values: • 1 – Normal • 3 – Flat Fee Monetary • 4 – Non-Billable
AdjustRateCode		char(30)	This is the corresponding rate code for adjustments.
LoadTrackingUID		int	The unique identifier that tracks the rate when loaded to the database from a location outside of SmartCloud Cost Management.
RateTable	PK[a]	char(30)	The name of the rate table that contains the rate code.
RateCode	PK[a]	char(30)	The code for the rate.
RateIndex		decimal(10,4)	The index number for the rate code.

Field Name	Key	Type	Field Description
RateValue		decimal(18,8)	The per unit value for the rate code.
ResourceValue		decimal(19,4)	Deprecated.
DiscountDollars		decimal(10,4)	Deprecated.
Discount		decimal(7,4)	Deprecated.
DiscountIndex		int	Deprecated.
Description		varchar(255)	A description of the rate code.
Factor		decimal(15,8)	A user-defined resource conversion factor value for the resource unit value in reports.
RateValue1		char(1)	Indicates whether 4 decimal positions are used in the rate value in reports: • F (4 digits are used) • Blank (4 digits are used [the default])
RateValue2		char(1)	Indicates whether the rate is per resource unit or per 1000 units in reports: • M (Per 1000 units) • Blank (Per unit)
RateValue3		char(1)	Indicates the resource conversion factor for the resource unit value in reports. • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.) • Blank (No conversion factor)
RateValue4		char(1)	Indicates whether the rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations)
RateValue5		char(1)	Indicates the number of decimal digits that are displayed in the resource units value in reports: • 0-5 (0 through 5 digits) • Blank (2 digits, the default)

Field Name	Key	Type	Field Description
RateValue6		char(1)	For future use.
RateValue7		char(1)	Indicates whether the resource units for the rate code are considered a monetary amount rather than units if utilization: • $ (flat fee) • Blank (not flat fee)
RateValue8		char(1)	User-defined report flag.
RateValue9		char(1)	For future use.
RateValue10		char(1)	User-defined report flag.
RateValue11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y (Yes) • N (No)
EffDate		datetime	The effective date for the rate code.
ExpiryDate		datetime	The end date for the rate code.
Comments		varchar(255)	Comments for the rate code.
DetailDescription		varchar(255)	User-defined description of the rate code for custom reports.
CurrencySymbol		varchar(3)	User-defined currency symbol for custom reports.
UnitCode		int	This is the code representing the units the rate code is measured in.
UnitDescription		varchar(300)	This is the first part of the description of a unit used for reporting.
UnitsDescription		varchar(300)	This is the second part of the description of a unit used for reporting.
LowerThreshold		decimal(18,8)	The lower threshold for the rate.
UpperThreshold		decimal(18,8)	The upper threshold for the rate.
PCT		decimal(6,3)	The percentage of units to be rated.
ThresholdDisplay		decimal(18,8)	Calculated threshold used in reports.
ThresholdSign		varchar(2)	Sign used to display the threshold in reports.
TierMethod		int	The tiering method for the rate. This can have the following values: • 1 – Non-Tiered • 2 – Individual • 3 – Highest • 4 – Individual Percent • 5 – Highest Percent

Field Name	Key	Type	Field Description
RateTypeCode		int	This is the type of rate code used for display purposes. This can have the following values: • 1 – Normal • 3 – Flat Fee Monetary • 4 – Non-Billable

a. PK = Primary Key

SCDIMRateShift Table

The SCDimRateShift table consolidates the information about rate groups, rates and rate shifts for use with Tivoli Common Reporting Cognos reports.

Field Name	Key	Type	Field Description
RateGroup	PK[a]	int	A sequential number assigned to the rate group.
ParentUID		int	For future use.
GroupTitle		varchar(300)	The rate group title.
GroupTitleLong		varchar(300)	A description of the rate group.
PrintIndex		int	The order in which the rate group is displayed in reports.
ParentLoadTrackingUID		int	The unique identifier that tracks the parent rate when loaded to the database from a location outside of SmartCloud Cost Management.
ParentRateCode		char(30)	The code for the parent rate.
ParentRateIndex		decimal(10,4)	The index number for the parent rate code.
ParentRateValue		decimal(18,8)	The per unit value for the parent rate code.
ParentResourceValue		decimal(19,4)	Deprecated
ParentDiscountDollars		decimal(10,4)	Deprecated
ParentDiscount		decimal(7,4)	Deprecated
ParentDiscountIndex		int	Deprecated
ParentDescription		varchar(255)	parent description
ParentFactor		decimal(15,8)	A user-defined resource conversion factor value for the resource unit value in reports used by the parent rate code.
ParentRateValue1		char(1)	Indicates whether 4 decimal positions are used in the rate value in reports for the parent rate code: • F (4 digits are used) • Blank (4 digits are used [the default])

Field Name	Key	Type	Field Description
ParentRateValue2		char(1)	Indicates whether the rate is per resource unit or per 1000 units in reports for the parent rate code: • M (Per 1000 units) • Blank (Per unit)
ParentRateValue3		char(1)	Indicates the resource conversion factor for the resource unit value in reports for the parent rate code. • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.) • Blank (No conversion factor)
ParentRateValue4		char(1)	Indicates whether the parent rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations)
ParentRateValue5		char(1)	Indicates the number of decimal digits that appear in the resource units value in reports for the parent rate code.: • 0-5 (0 through 5 digits) • Blank (2 digits, the default)
ParentRateValue6		char(1)	For future use.
ParentRateValue7		char(1)	Indicates whether the resource units for the rate code are considered a monetary amount rather than units if utilization: • $ (flat fee) • Blank (not flat fee)
ParentRateValue8		char(1)	User-defined report flag.
ParentRateValue9		char(1)	For future use.
ParentRateValue10		char(1)	User-defined report flag.
ParentRateValue11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y– yes • N –no

Field Name	Key	Type	Field Description
ParentComments		varchar(255)	Comments for the parent rate code.
ParentDetailDescription		varchar(255)	User-defined description of the rate code for custom reports.
ParentCurrencySymbol		varchar(3)	User-defined currency symbol for custom reports used by the parent rate code.
ParentRateTypeCode		int	This is the type of parent rate code used for display purposes.
ParentTierMethod		int	The tiering method for the parent rate. This can have the following values: • 0 - No tiering • 1 - Individual • 2 - highest
ParentTierMethod		int	The tiering method for the parent rate. This can have the following values: • 1 – Non-Tiered • 2 – Individual • 3 – Highest • 4 – Individual Percent • 5 – Highest Percent
ParentRateTypeCode		int	This is the type of rate code used for display purposes. This can have the following values: • 1 – Normal • 3 – Flat Fee Monetary • 4 – Non-Billable
AdjustRateCode		char(30)	This is the corresponding rate code for adjustments.
LoadTrackingUID		int	The unique identifier that tracks the rate when loaded to the database from a location outside of SmartCloud Cost Management.
RateTable	PK[a]	char(30)	The name of the rate table that contains the rate code.
RateCode	PK[a]	char(30)	The code for the rate.
RateIndex		decimal(10,4)	The index number for the rate code.
RateRateValue		decimal(18,8)	The per unit value for the rate code.
ResourceValue		decimal(19,4)	Deprecated.
DiscountDollars		decimal(10,4)	Deprecated.
Discount		decimal(7,4)	Deprecated.
DiscountIndex		int	Deprecated.
RateDescription		varchar(255)	A description of the rate code.

Field Name	Key	Type	Field Description
Factor		decimal(15,8)	A user-defined resource conversion factor value for the resource unit value in reports.
RateRateValue1		char(1)	Indicates whether 4 decimal positions are used in the rate value in reports: • F (4 digits are used) • Blank (8 digits are used [the default])
RateRateValue2		char(1)	Indicates whether the rate is per resource unit or per 1000 units in reports: • M (Per 1000 units) • Blank (Per unit)
RateRateValue3		char(1)	Indicates the resource conversion factor for the resource unit value in reports. • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.) • Blank (No conversion factor)
RateRateValue4		char(1)	Indicates whether the rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations)
RateRateValue5		char(1)	Indicates the number of decimal digits that appear in the resource units value in reports: • 0-5 (0 through 5 digits) • Blank (2 digits, the default)
RateRateValue6		char(1)	For future use.
RateRateValue7		char(1)	Indicates the whether the resource units for the rate code are considered a monetary amount rather than units if utilization: • $ (flat fee) • Blank (not flat fee)
RateRateValue8		char(1)	User-defined report flag.

Field Name	Key	Type	Field Description
RateRateValue9		char(1)	For future use.
RateRateValue10		char(1)	User-defined report flag.
RateRateValue11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y (Yes) • N (No)
RateCodeEffDate		datetime(10)	The effective date for the rate code.
RateCodeExpiryDate		datetime(10)	The expiry date for the rate code.
RateComments		varchar(255)	Comments for the rate code.
DetailDescription		varchar(255)	User-defined description of the rate code for custom reports.
CurrencySymbol		varchar(3)	User-defined currency symbol for custom reports.
UnitCode		int	This is the code representing the units the rate code is measured in.
UnitDescription		varchar(300)	This is the first part of the description of a unit used for reporting.
UnitsDescription		varchar(300)	This is the second part of the description of a unit used for reporting.
ShiftCode	PK[a]	char(1)	The shift code.
RateValue		decimal(18,8)	The rate value for the shift.
Description		varchar(255)	A description of the shift (Swing, Night, and so on).
Comments		varchar(255)	For future use.
LowerThreshold		decimal(18,8)	The lower threshold for the rate.
UpperThreshold		decimal(18,8)	The upper threshold for the rate.
PCT		decimal(6,3)	The percentage of units to be rated.
ThresholdDisplay		decimal(18,8)	Calculated threshold used in reports.
ThresholdSign		varchar(2)	Sign used to display the threshold in reports.
TierMethod		int	The tiering method for the rate. This can have the following values: • 1 – Non-Tiered • 2 – Individual • 3 – Highest • 4 – Individual Percent • 5 – Highest Percent

Field Name	Key	Type	Field Description
RateTypeCode		int	This is the type of rate code used for display purposes. This can have the following values: • 1 – Normal • 3 – Flat Fee Monetary • 4 – Non-Billable

a. PK = Primary Key

SCRate Table

The SCRate table stores the rate codes and rates for resource usage.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	The unique identifier that tracks the rate when loaded to the database from a location outside of SmartCloud Cost Management.
RateCode	PK[a]	char(30)	The code for the rate.
RateIndex		int	The index number for the rate code.
RateValue		decimal(18,8)	The per unit value for the rate code.
ResourceValue		decimal(19,4)	Deprecated.
DiscountDollars		decimal(19,4)	Deprecated.
Discount		decimal(7,4)	Deprecated.
DiscountIndex		int	Deprecated.
Factor		decimal(15,8)	A user-defined resource conversion factor value for the resource unit value in reports.
RateTable	PK[a]	char(30)	The name of the rate table that contains the rate code.
EffDate	PK[a]	datetime	The effective date for the rate code.
ExpiryDate		datetime	The end date for the rate code.
RateValue1		char(1)	Indicates whether four decimal positions are used in the rate value in reports: • F (Four digits are used) • Blank (Eight digits are used [the default])
RateValue2		char(1)	Indicates whether the rate is per resource unit or per thousand units in reports: • M (Per thousand units) • Blank (Per unit)

Field Name	Key	Type	Field Description
RateValue3		char(1)	Indicates the resource conversion factor for the resource unit value in reports. • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.) • Blank (No conversion factor)
RateValue4		char(1)	Indicates whether the rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations)
RateValue5		char(1)	Indicates the number of decimal digits that appear in the resource units value in reports: • 0-5 (0 through 5 digits) • Blank (2 digits, the default)
RateValue6		char(1)	Indicates ⌂whether the resource units for the rate code should be averaged.
RateValue7		char(1)	Indicates whether the resource units for the rate code are considered a monetary amount rather than units of utilization: • $ (flat fee) • Blank (not flat fee)
RateValue8		char(1)	User-defined report flag.
RateValue9		char(1)	For future use.
RateValue10		char(1)	User-defined report flag.
RateValue11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y (Yes) • N (No)
Comments		varchar(255)	Comments for the rate code.
DetailDescription		varchar(255)	User-defined description of the rate code for custom reports.
CurrencySymbol		varchar(3)	User-defined currency symbol for custom reports.

Field Name	Key	Type	Field Description
Threshold		decimal(18,8)	This is the threshold for the child rate code.
UnitCode		int	This is the code representing the units the rate code is measured in.
AdjustRateCode		char(30)	This is the corresponding rate code for adjustments.
ParentRateCode		char(30)	This is the rate code for the parent rate code.
RateGroup		int	A sequential number assigned to the rate group.
PCT		decimal(6,3)	The percentage of units to be used.
TierMethod		int	The tiering method for the rate. This can have the following values: • 1 – Non-Tiered • 2 – Individual • 3 – Highest • 4 – Individual Percent • 5 – Highest Percent
RateTypeCode		int	This is the type of rate code used for display purposes. This can have the following values: • 1 – Normal • 3 – Flat Fee Monetary • 4 – Non-Billable

a. PK = Primary Key

SCRateTable Table

The SCRateTable table is the rate table for the rates. Rate tables are used to perform differential costing, for example, you can charge Client A different rates than Client B.

Field Name	Key	Type	Field Description
RateTable	PK[a]	char(30)	The name of the rate table.
Description		varchar(255)	A description of the rate table.
EffDate		datetime	The effective date for the rate table.
ExpiryDate		datetime	The expiry date for the rate table.
LockDown		varchar(30)	Lockdown date.
CurrencySymbol		varchar(3)	Currency symbol.

a. PK = Primary Key

SCRateShift Table

The SCRateShift table defines the optional rate shifts for rate codes.

Field Name	Key	Type	Field Description
RateCode	PK[a]	char(30)	The rate code.
RateCodeEffDate		datetime	The effective date for the rate code.
ShiftCode	PK[a]	char(1)	The shift code.
RateValue		decimal(18,8)	The rate value for the shift.
Description		varchar(255)	A description of the shift (Swing, Night, etc.).
Comments		varchar(255)	For future use.
RateTable		char(30)	The code used to identify the rate table.

a. PK = Primary Key

SCResourceUtilization Table

The SCResourceUtilization table stores the resource information contained in the Resource file. The Resource file is an optional file produced by the Bill program.

Data from the SCResourceUtilization table is not used in the standard SmartCloud Cost Management reports.

In most cases, you will not need to load the data in SCAcct Detail file to the database.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	The load tracking unique identifier from the SCLoadTracking table.
DetailUID		int	The unique identifier that tracks the Resource load.
DetailLine		int	The number of the record in the input CSR or CSR+ file that contains the resource (i.e., 1 if it is the first record, 2 if it is the second record, etc.)
AccountCode		char(127)	The account code.
Aggregate		int	A count of the number of original input CSR or CSR+ records that have been aggregated into this record.
StartDate		datetime	The resource usage start date.
EndDate		datetime	The resource usage end date.
ShiftCode		char(1)	The resource usage shift code.
AuditCode		varchar(255)	The audit code (if applicable).
SourceSystem		char(8)	The system identifier for this resource.
RateCode		char(30)	The rate code of this resource.

Field Name	Key	Type	Field Description
ResourceUnits		decimal(18,5)	The quantity of the resource used.
RateTable		char(30)	The name of the rate table.
RateCodeEffDate		datetime	The effective date for the rate code.

SCSummaryDaily Table

The SCSummaryDaily table stores the data in the SCSummary table at a monthly rather than daily level.

The fields are the same as the SCSummary table.

SCSummary Table

The SCSummary table stores the information contained in the Summary file. The Summary file is produced by the Bill program.

This table contains both resource utilization and cost data for reports. Note the BillFlag fields reflect the flags that were set in the SCRate table at the time that CIMSBill was run.

Field Name	Key	Type	Field Description
LoadTrackingUID		int	The load tracking unique identifier from the SCLoadTracking table.
Year		int	The resource usage year.
Period		int	The resource usage period
Shift		char(1)	The resource usage shift code.
AccountCode		char(127)	The account code.
LenLevel1		int	Values are the following: • 0-The record was created from a Detail record. • 1-The record was by created by reprocessing a Summary record. • 2-The record was by created by processing a Summary record twice. If the Summary record was processed more than twice, the value would be 3, 4, 5, etc.
LenLevel2		int	A value of 0.
LenLevel3		int	A value of 0.
LenLevel4		int	A value of 0.
RateTable		char(30)	The name of the rate table that contains the rate code.
RateCode		char(30)	The rate code for the resource.
StartDate		datetime	The accounting start date.
EndDate		datetime	The accounting end date.

Field Name	Key	Type	Field Description
BillFlag1		char(1)	Indicates whether four decimal positions are used in the rate value in reports: • F (Four digits are used) • Blank (Eight digits are used [the default])
BillFlag2		char(1)	Indicates whether the rate is per resource unit or per thousand units in reports: • M (Per thousand units) • Blank (Per unit)
BillFlag3		char(1)	Indicates the resource conversion factor for the resource unit value in reports: • 1 (Divide total resource value by 60) • 2 (Divide total resource value by 3600) • 3 (Divide total resource value by 1000) • 4 (Multiply total resource value by 60) • 5 (Divide total resource value by 60000) • # (Multiple total resource value by user-defined number in Factor field.)
BillFlag4		char(1)	Indicates whether the rate code is included in zero cost calculations: • N (The rate will not be included in zero cost calculations) • Blank (The rate will be included in zero cost calculations) For more information about zero cost calculations, see page 3-11.
BillFlag5		char(1)	Indicates the number of decimal digits that appear in the resource units value in reports: • 1-5 (1 through 5 digits) • Blank (2 digits, the default)
BillFlag6		char(1)	For future use.
BillFlag7		char(1)	Indicates the whether the resource units for the rate code are considered a monetary amount rather than units if utilization: • $ (flat fee) • Blank (not flat fee)
BillFlag8		char(1)	User-defined report flag.

Field Name	Key	Type	Field Description
BillFlag9		char(1)	For future use.
RateValue		decimal(18,8)	Per unit charge for this rate.
ResourceUnits		decimal(18,5)	The total units for this rate.
BreakId		int	Deprecated.
MoneyValue		decimal(19,6)	The total currency units for this rate. This will be 21,6 only if using an existing Oracle database prior to 4.2 and then upgrading to 4.2 or later.
UsageStartDate		datetime	The resource usage start date.
UsageEndDate		datetime	The resource usage end date.
RunDate		datetime	The date and time that CIMSBill was run.
BillFlag10		char(1)	User-defined report flag.
BillFlag11		char(1)	Indicates whether the rate should be included in CPU normalization: • Y (Yes) • N (No)
UnitCode		int	This is the code representing the units the Rate code is measured in.
RateCodeEffDate		datetime	The effective date for the rate code.
UsageYear		int	The resource usage year.
UsagePeriod		int	The resource usage period.

SCUnits Table

The SCUnits table stores the units that the rate code is measured in.

Field Name	Key	Type	Field Description
UnitCode	PK[a]	int	The code representing the units the rate code is measured in.
UnitDescription		varchar(50)	This is the first part of the description of a unit used for reporting.
UnitsDescription		varchar(50)	This is the second part of the description of a unit used for reporting.

SmartCloud Cost Management files

This section describes the IBM SmartCloud Cost Management input and output files.

Note: The format of the Resource, Detail and Summary files has changed in the IBM SmartCloud Cost Management 2.1.0.5 release. Older versions of these files continue to be supported in the Processing Engine, with the caveat that Rates referred to in the Resource, Detail or Summary files will be assumed to be from the "STANDARD" rate table and have an effective date of "1980-01-01".

CSR file

SmartCloud Cost Management processes and applies business rules to resource usage data from any application, system, or operating system on any platform. The primary method for loading this data into SmartCloud Cost Management is the CSR or CSR+ file. This topic describes the file layout of the CSR file.

CSR records are in a standard ASCII display format (no packed, binary, or bit data) with commas for delimiters and decimal points included in resource amounts. A negative sign should precede negative numeric data, with no sign when the data is positive. When the identifier data contains commas, there must be double quotes around the identifier character data.

CSR records can be up to 32,000 bytes long and can contain a very large number of identifiers and resources. However, the maximum length for the records in the output Detail file is 5,000 bytes with a limit of 100 resources.

The following table describes the fields in the CSR record.

Pos.	Field Name	Length	Type	Description
1	Record ID	8	Character	Defines the source of data. For example, storage data from the CIMSWinDisk data collector contains a record ID of WinDisk. There is no standard for this ID and any unique combination of characters can be used.
2	Start Date of Usage	8	Number	Date in format YYYYMMDD.
3	End Date of Usage	8	Number	Date in format YYYYMMDD.
4	Start Time of Usage	8	Character	Time in format HH:MM:SS.
5	End Time of Usage	8	Character	Time in format HH:MM:SS.
6	Shift Code	1	Character	Alphanumeric code denoting time of day usage occurred. Allows billing different rates by shift. If you do not want to charge by shift, the field should be blank.

Pos.	Field Name	Length	Type	Description
7	Number of Identifiers	2	Number	Number of identifiers in the following fields.
8	Identifier Name 1	32	Character	The name of the identifier.
9	Identifier Value 1	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
10	Identifier Name 2	32	Character	The name of the identifier.
11	Identifier Value 2	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
12	Identifier Name x	32	Character	The name of the identifier.
13	Identifier Value x	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
X	Number of Resources	2	Number	Number of resources being tracked in the following fields.
X	Rate Code 1	8	Character	The rate code for the resource.
X	Resource Value 1	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.
X	Rate Code 2	8	Character	The rate code for the resource.
X	Resource Value 2	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.
X	Rate Code x	8	Character	The rate code for the resource.

Pos.	Field Name	Length	Type	Description
X	Resource Value x	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.

CSR+ file

The CSR+ file is used to input data into SmartCloud Cost Management and Accounting Manager. This topic describes the file layout of the CSR+ file.

The format of the CSR+ file record is the same as the CSR record with the exception that the CSR+ record contains a header at the beginning of the record. This is a fixed header that enables the records to be sorted more easily. The header is in the following format:

```
"CSR+<usage start date><usage end date><account code length><account code><x'40'>",
```

Note that the quotation marks and comma are included in the header.

Examples

```
"CSR+2007022820070228010aaaaaaaaa ",S390DB2,20070228,...
"CSR+2007022820070228010bbbbbbbbb ",S390DB2,20070228,...
```

In these examples, the usage start and end dates are February 28, 2007 (20070228). The account codes aaaaaaaaa and bbbbbbbbb are 10 characters. The account codes are followed by a space (x'40'). The information after the header (S90DB2,20070228,...) represents the remaining fields found in the record. These fields are described in the following table.

Note: The CSR+ record does not supersede the CSR record. SmartCloud Cost Management uses both record types.

Pos.	Field Name	Length	Type	Description
1	Header	Variable (26 to 153 characters)	Character	A fixed header used for sorting.
2	Record ID	8	Character	Defines the source of data. For example, storage data from the CIMSWinDisk collector contains a record ID of WinDisk. There is no standard for this ID and any unique combination of characters can be used.
3	Start Date of Usage	8	Number	Date in format YYYYMMDD.
4	End Date of Usage	8	Number	Date in format YYYYMMDD.

Pos.	Field Name	Length	Type	Description
5	Start Time of Usage	8	Character	Time in format HH:MM:SS.
6	End Time of Usage	8	Character	Time in format HH:MM:SS.
7	Shift Code	1	Character	Alphanumeric code denoting time of day usage occurred. Allows billing different rates by shift. If you do not want to charge by shift, the field should be blank.
8	Number of Identifiers	2	Number	Number of identifiers in the following fields.
9	Identifier Name 1	32	Character	The name of the identifier.
10	Identifier Value 1	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
11	Identifier Name 2	32	Character	The name of the identifier.
12	Identifier Value 2	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
13	Identifier Name x	32	Character	The name of the identifier.

Pos.	Field Name	Length	Type	Description
14	Identifier Value x	Variable (Maximum 255)	Character	Includes items such as database name, server name, LAN ID, user ID, program name, region, system ID, and so forth. This should be shortened as much as possible to a meaningful code for further translation.
X	Number of Resources	2	Number	Number of resources being tracked in the following fields.
X	Rate Code 1	8	Character	The rate code for the resource.
X	Resource Value 1	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.
X	Rate Code 2	8	Character	The rate code for the resource.
X	Resource Value 2	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.
X	Rate Code x	8	Character	The rate code for the resource.
X	Resource Value x	Variable	Number	Resource usage value such as CPU time, Input/Outputs, megabytes used, lines printed, transactions processed, etc.

Ident file

The Ident file contains all the identifiers (such as user ID, jobname, department code, server name, etc.) that are contained in the input records. These identifiers are used during account code conversion to create your target account code structure. The Ident file is created by the Bill program.

The Ident file is a simple, comma-delimited file. The following is the file layout for the Ident file.

Field	Description
Unique Load ID	The unique ID for the load.
Record Number	The record number.
Identifier Name	The name of the identifier (e.g., Jobname).
Identifier Value	The value for the identifier (e.g., ACPSJEFU).

Resource file

The Resource file is basically the same as the Detail file but without CPU normalization, proration, or include/exclude processing applied to the resources. The Resource file is optional and is created by the Bill program.

The following is the file layout for the Resource file.

Field	Starting Position	Length	Description
DETAIL-REC-TYPE	1	3	Always '991'.
DETAIL-REC-ID	5	8	Identifies the type of record. For example: OS390DB2 - (OS/390® DB2 records)
DETAIL-EYE-CATCH	14	7	The version of the record.
DETAIL-LOAD-ID	22	10	The unique ID of the file that contained this detail record.
DETAIL-REC-NUMBER	33	10	The record number within the original detail file.
DETAIL-NUM-RECS	44	10	The number of records that were aggregated to make this one record. This field applies only to mainframe data.
DETAIL-SORT-ID	55	1	(Reserved)
DETAIL-SYSTEM-ID	57	32	The system ID of the source of the record.
DETAIL-WORK-ID	90	32	The work id where the record came from (could be subsystem name, could be Oracle instance name).
DETAIL-START-DATE	123	8	The start date of the record.
DETAIL-END-DATE	132	8	The end date of the record.
DETAIL-START-TIME	141	8	The start time of the record.
DETAIL-END-TIME	150	8	The end time of the record.
DETAIL-SHIFT	159	1	The shift code.
DETAIL-DOW	161	1	The day of week.
DETAIL-ACCOUNT-CODE	163	128	The account code.

Field	Starting Position	Length	Description
DETAIL-AUDIT-CODE	292	8	The audit code.
DETAIL-INCLEXCL-ARE A	301	60	Include/exclude data range.
DETAIL-RES-NUMBER	362	2	Number of resources being tracked in the following fields.
DETAIL-RES-INFO	365	x	Occurs 1 to 100 times depending DETAIL-RES-NUMBER (see preceding).
DETAIL-RATE-CODE			The resources rate code.
DETAIL-RATE-TABLE		30	Rate table of rate.
DETAIL-RATE-EFFECTIVEDATE		8	Effective date of rate.
DETAIL-RESOURCE-VAL			The resource value.
DETAIL-RESOURCE-SIGN			This field is blank if the resource is positive and '-' if the resource is negative.

Detail file

The Detail file is created by the Bill program.

The Detail file differs from the Resource file in that the Detail file reflects any proration, CPU normalization, or include/exclude processing that was performed. The Detail file also includes accounting dates.

The following is the file layout for the Detail file.

Field	Starting Position	Length	Description
DETAIL-REC-TYPE	1	3	Always '991'.
DETAIL-REC-ID	5	8	Identifies the type of record. For example: OS390DB2 - (OS/390 DB2 records)
DETAIL-EYE-CATCH	14	7	The version of the record.
DETAIL-LOAD-ID	22	10	The unique ID of the file that contained this detail record.
DETAIL-REC-NUMBER	33	10	The record number within the original detail file.
DETAIL-NUM-RECS	44	10	The number of records that were aggregated to make this one record. This field applies only to mainframe data.
DETAIL-SORT-ID	55	1	(Reserved)
DETAIL-SYSTEM-ID	57	32	The system ID of the source of the record.
DETAIL-WORK-ID	90	32	The work id where the record came from (could be subsystem name, could be oracle instance name).

Field	Starting Position	Length	Description
DETAIL-START-DATE	123	8	The start date of the record.
DETAIL-END-DATE	132	8	The end date of the record.
DETAIL-START-TIME	141	8	The start time of the record.
DETAIL-END-TIME	150	8	The end time of the record.
ACCOUNTING-START-DA TE	159	8	The accounting period start date.
ACCOUNTING-END-DATE	168	8	The accounting period end date.
DETAIL-SHIFT	177	1	The shift code.
DETAIL-DOW	179	1	The day of week.
DETAIL-ACCOUNT-CODE	181	128	The account code.
DETAIL-AUDIT-CODE	310	8	The audit code.
DETAIL-INCLEXCL-AREA	319	60	Include/exclude data range.
DETAIL-RES-NUMBER	380	2	Number of resources being tracked in the following fields.
DETAIL-RES-INFO	383	x	Occurs 1 to 100 times depending detail-res-number (see above).
DETAIL-RATE-CODE		30	The resources rate code.
DETAIL-RATE-TABLE		30	Rate Table of Rate.
DETAIL-RATE-EFFECTIVEDATE		8	Effective date of Rate.
DETAIL-RATE-UNIT		4	Unit type of Rate.
DETAIL-RATE-VALUE		30	The resources rate value.
DETAIL-MONEY-VALUE		30	The total charge for the resource units after the rate value is applied.
DETAIL-RESOURCE-VAL		30	The resource value.
DETAIL-RESOURCE-SIGN			This field is blank if the resource is positive and '-' if the resource is negative.

Summary file

This Summary file provides resource usage and cost data used for Web reports or for input to other financial or resource accounting systems. The Summary file is created by the Bill program.

The following is the file layout for the Summary file.

Field	Start Position	Length	Type
"SUMMARY"	1	8	Character
Reserved	9	3	Numeric
Reserved	12	3	Numeric
Reserved	15	3	Numeric
Reserved	18	3	Numeric
AccountCode	21	128	Character
RateTable	149	30	Character
SourceSystem	179	1	Character
RateCode	180	30	Character
RateCodeEffDate	210	8	Numeric
ShiftCode	218	1	Numeric
UnitCode	219	4	Numeric
AccountingFromDate	223	8	Numeric
AccountingToDate	231	8	Numeric
BillFlag1	239	1	Character
BillFlag2	240	1	Character
BillFlag3	241	1	Character
BillFlag4	242	1	Character
BillFlag5	243	1	Character
BillFlag6	244	1	Character
BillFlag7	245	1	Character
BillFlag8	246	1	Character
BillFlag9	247	1	Character
BillFlag10	248	1	Character
BillFlag11	249	1	Character
RateValue	250	18	Numeric
ResourceUnits	268	18	Numeric
MoneyValue	286	18	Numeric
BreakId	304	1	Character
Conv Factor	305	13	Numeric
Release ID	318	6	Numeric
Date-Century	324	2	Numeric
Date-Year	326	2	Numeric
Date-Month	328	2	Numeric
Date-Day	330	2	Numeric

Field	Start Position	Length	Type
Time-HH	332	2	Numeric
Time-MM	334	2	Numeric
Time-SS	336	2	Numeric
Period	338	2	Numeric
Year	340	4	Numeric
UsageStartDate	344	8	Numeric
UsageEndDate	352	8	Numeric
UsagePeriod	360	2	Numeric
UsageYear	362	4	Numeric

Accessibility features for SmartCloud Cost Management

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features of SmartCloud Cost Management are described in this topic.

Accessibility features

The following list includes the major accessibility features in SmartCloud Cost Management:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

Note: The default configuration of JAWS screen reader does not read tooltips. JAWS users must enable their current mode to read tooltips by selecting **Utilities > Settings Center > Speech Verbosity > Verbosity Level > Configure Verbosity Levels**.

User documentation is provided in HTML and PDF format. Descriptive text is provided for all documentation images.

The knowledge center, and its related publications, are accessibility-enabled.

Related accessibility information

You can view the publications for SmartCloud Cost Management in Adobe Portable Document Format (PDF) using the Adobe Reader. PDF versions of the documentation are available in the knowledge center.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

© Copyright IBM Corp. 2013, 2015

Notices

This information was developed for products and services that are offered in the USA.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 may 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those

© Copyright IBM Corp. 2013, 2015

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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 measurements 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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 2013, 2015. All rights reserved.

Programming interface information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of IBM SmartCloud Cost Management. This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM SmartCloud Cost Management. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking: *Programming Interface information*.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at www.ibm.com/legal/copytrade.shtml.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name, or other personally identifiable information for purposes of session management, enhanced user usability, single sign-on configuration. These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Glossary

This glossary includes terms and definitions for IBM Cloud Orchestrator.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

account code
A code that uniquely identifies an individual, billing, or reporting entity within chargeback and resource accounting.

account code conversion table
An ASCII text file that contains the definitions that are required to convert the identifier values defined by the account code input field to the user-defined output account codes.

account report
A report that is used to show account level information for usage and charge.

availability zone
A logical group of OpenStack Compute hosts. It provides a form of physical isolation and redundancy from other availability zones, such as by using separate power supply or network equipment.

© Copyright IBM Corp. 2013, 2015

B

Bill program
A program that performs cost extensions within SmartCloud Cost Management and summarizes cost and resource utilization by account code. The Bill program uses the rate code table that is assigned to the client to determine the amount to be charged for each resource consumed.

building block
The model of an image that is created by combining models of a base operating system and software bundles. Each building block contains a semantic and functional model that describes the contents of the components, for example, the installed products, supported operating systems, prerequisites, and requirements.

business object
A software entity that represents a business entity, such as an invoice. A business object includes persistent and nonpersistent attributes, actions that can be performed on the business object, and rules that the business object is governed by.

business process
A defined set of business activities that represent the required steps to achieve a business objective. A business process includes the flow and use of information and resources.

C

chargeback identifier
A label, which is often tied to an algorithm or set of rules, that is not guaranteed to be unique, but is used to identify and distinguish a specific chargeback item or chargeback entity from others.

compute node
A node that runs a virtual machine instance, which provides a wide range of services, such as providing a development environment or performing analytics.

consolidation process
A process during which the data collectors process the nightly accounting and storage files that were created by the data collection scripts and produce an output CSR file.

conversion mapping
An entry in a mapping table which allows you to map identifiers to accounts or other identifiers.

custom node
A virtual image part that provides an unconfigured node for a pattern that has a deployment manager or a control node as its base.

E

exception file
A file that contains a list of records with identifier names that do not have a matching Parameter IdentifierName attribute value.

exception processing
A process in which the system writes all records that to do match an entry in the account code conversion table to an exception file.

H

human service
An activity in the business process definition that creates an interactive task that the process participants can perform in a web-based user interface.

hypervisor
Software or a physical device that enables multiple instances of operating systems to run simultaneously on the same hardware.

K

kernel The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

P

parameter (parm)
A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

parm See parameter.

performance counter
A utility that provides a way for software to monitor and measure processor performance.

primary key
In a relational database, a key that uniquely identifies one row of a database table.

process application
A container in the Process Center repository for process models and supporting implementations. A process application typically includes business process definitions (BPDs), the services to handle implementation of activities and integration with other systems, and any other items that are required to run the processes. Each process application can include one or more tracks.

proration
A process that distributes the overall or individual resources of an account and the cost of those resources across multiple accounts at a specified percentage.

proration table
An ASCII text file that defines the identifier values and rate codes that are used in the proration process.

R

rate code
The identifier of a rate that is used to link a resource unit or volume metric with its charging characteristics.

rate group
A group of rate codes that is used to create rate subtotals in reports, graphs, and spreadsheets.

registry
A repository that contains access and configuration information for users, systems, and software.

S

service operation
A custom operation that can be run in the context of the data center. These operations are typically administrative operations and are used to automate the configuration. Service operations can also be used to enhance the catalog of available services with extra functionality.

software bundle
A collection of software installation files, configuration files, and metadata that can be deployed on a virtual machine instance.

T

toolkit
A container where artifacts can be stored for reuse by process applications or other toolkits.

V

virtual machine (VM)
An instance of a data-processing system that appears to be at the exclusive disposal of a single user, but whose functions are accomplished by sharing the resources of a physical data-processing system.

VM See virtual machine.



Product Number: 5725-H28

Printed in USA