AIX Version 6.1

Advanced Accounting Subsystem

SC23-6606-00

AIX Version 6.1

Advanced Accounting Subsystem

SC23-6606-00

Note
Before using this information and the product it supports, read the information in “Notices,” on page 45.

First Edition (November 2007)

This edition applies to AIX Version 6.1 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6C006, 11501 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About this document

This document provides system administrators with conceptual and procedural information about how to set up, administer, and manage the Advanced Accounting subsystem. Information about projects, policies, transactional accounting, interval accounting, and data aggregation is included. This information is also available on the documentation CD that is shipped with the operating system.

How to use this document

This document is organized to provide overview information and, when needed, conceptual details for major elements of the AIX operating system. Only general descriptions of basic tasks are provided. For task instructions, see the *Operating system and device management*.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Case-sensitivity in AIX

Everything in the AIX® operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type `LS`, the system responds that the command is `not found`. Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related publications

- *Operating system and device management*
- *Security*
- *Installation and migration*
- *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*
- *AIX Version 6.1 Kernel Extensions and Device Support Programming Concepts*
- *AIX Version 6.1 Files Reference*
- *Performance management*
- *AIX Version 6.1 Commands Reference, Volume 1*
- *AIX Version 6.1 Commands Reference, Volume 4*
- *AIX Version 6.1 Commands Reference, Volume 5*

Advanced Accounting subsystem

This topic provides system administrators with conceptual and procedural information about how to set up, administer, and manage Advanced Accounting. Information about projects, policies, transactional accounting, interval accounting, and data aggregation is included.

To view or download the PDF version of this topic, click Advanced Accounting subsystem.

Introduction to the Advanced Accounting Subsystem

The Advanced Accounting subsystem, hereafter referred to as Advanced Accounting, is based on mainframe technology and features interval accounting, data aggregation, and dynamic classification of accounting data. You can customize Advanced Accounting for different computing environments. You can configure Advanced Accounting to produce the specific types of records needed for billing applications.

Advanced Accounting provides usage-based information for a wide variety of system resources so that you can develop comprehensive charge-back strategies. You can collect accounting data on resources such as disks, network interfaces, virtual devices, file systems, processors, and memory. Interval accounting gives you the ability to view this data over system administrator-defined time intervals in order to develop chronological views. This has several potential applications, including capacity planning.

Advanced Accounting also provides new statistics from previous accounting tools. For example, the process record provides microsecond-level CPU times, a memory integral based on elapsed time (standard UNIX® accounting bases it on CPU times), local and distributed logical file I/O, and local and remote socket I/O.

Interval accounting can be used to periodically record accounting data, enabling the production of more accurate bills. You can configure Advanced Accounting to produce intermediate process records for active processes. These records can be added to the completed process records to produce a bill that reflects the total use of system resources.

Data aggregation is a way to control the amount of data written to a data file. This helps system performance by reducing the overall resource load needed to run Advanced Accounting. Aggregation minimizes a system's I/O requirements, adding accounting records so that fewer records are written to the accounting file. It is transparent to applications and middleware.

Policies are rules that provide for the automatic classification of processes. Classification is done according to users, groups, and applications, categorizing the use of system resources by billable entities. These categories are called *projects*.

APIs are provided so that applications and middleware can describe the transactional nature of their workloads, enabling charge-backs for server processes. These APIs are intended to define and delineate transactions and to identify the end user, if possible. Advanced Accounting measures the resource use of transactions, if possible, and records all of this information in the accounting file.

Data files

Every statistic recorded by Advanced Accounting is written to a data file. When the data file reaches its capacity, the billing application can process the file. After it is processed, that file can be reused, and the cycle repeats.

You must create your accounting data files in order to begin collecting accounting statistics. This involves assessing your company's needs and determining how much data you will be collecting. You can start

Advanced Accounting and let it run for a period of time to see how much data is produced, giving you an idea of how much disk space to reserve for accounting, as well as how many files you will need to manage your data.

Although not required, it is a good idea to specify at least two data files so that Advanced Accounting can remain active at all times. Advanced Accounting writes to only one file at a time, but as it does, it requires exclusive access to this file. With two files, Advanced Accounting can write to one file while the other file is processed by the billing application.

Advance accounting data files support Partition Mobility. Before a migration, on the source partition, all accounting data is flushed to the accounting file. The file is closed once the data flush is complete. After the migration, a new accounting file is opened on the destination partition if a file is available.

Data file life cycle

Each data file goes through a life cycle as it moves through the accounting process.

The following process explains the life cycle of a data file:

1. A pool of pre-allocated data files are created and registered with Advanced Accounting.
2. As accounting statistics are collected from the system, data is written to the active data file.
3. The size of the data file (predetermined when the file is created) limits the amount of data that can be written to the file. Once the data file is full, no more data can be written to it. Advanced Accounting then switches to the next empty data file and begin writing data to it.
4. Once the data file is full, you can then use post-processing tools, such as a billing application, to process the accounting data into statistics for billing purposes.
5. After the data is extracted from the data file and processed, the file is reset so that it can be reused.

Creating a data file

You can create a data file to contain statistics recorded by Advanced Accounting.

This scenario describes how to create a data file using either the System Management Interface Tool (SMIT) or the command line. You can also use Web-based System Manager to create data files.

Things to Consider

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Change to root user.
2. On a command line, type: `smit create_aacct_file`.
3. In the **Accounting Data File** field, type the name of the file you want to create. You must include the fully qualified path name.
4. In the **Pre-allocated size** field, type the size (in megabytes) of the file you wish to create.
5. Press Enter to save the settings and create the data file.

As an example, to create a 2 MB data file named **testfile** from the command line, type `acctctl fadd /var/aacct/testfile 2`.

Data file management

You can use the command line and SMIT to manage your data files.

The following table shows the commands and SMIT fast paths available for managing accounting data files.

Table 1. Data file management commands

Task	Command	SMIT fast path
Allocate and define an accounting file with specified file name and size. The default size is in megabytes.	**acctctl fadd** *file size*	**smit create_aacct_file**
Remove the specified accounting file from the accounting subsystem. This will not remove the file from the file system.	**acctctl frm** *file*	**smit release_aacct_file**
Indicate that the specified file can be reused by the accounting subsystem.	**acctctl freset** *file*	**smit reuse_aacct_file**
Query the state and current use of the specified file. If no file is specified, all files are queried.	**acctctl fquery** [*file*]	**smit list_aacct_file**
Force Advanced Accounting to switch to a new accounting data file. The new file may be optionally specified.	**acctctl fswitch** [*file*]	**smit switch_acct_file**

For more information, see *AIX Version 6.1 Commands Reference, Volume 1*.

Notification messages

Advanced Accounting continuously writes accounting data to registered accounting data files. It is necessary to monitor the state of these files to ensure that Advanced Accounting always has a place to record data.

Advanced Accounting produces messages to monitor the state of data files and the subsystem status. Use messages related to the data files to trigger actions based on events as they occur. A message is produced when a file is 90% full and when it is 100% full, indicating that administrative action is needed. The **acctctl** command is used to manage accounting data files. For additional information on the acctctl command, see **acctctl**.

You can use the **cron** facility to periodically execute a shell script or command that examines and manages accounting data files. By default, AIX records information about the Advanced Accounting subsystem using the **syslog** daemon. For more information about the **syslog** daemon, see *AIX Version 6.1 Commands Reference, Volume 5*.

The **readaacct** command may be used to extract accounting records from accounting data files. It is a sample command, so be aware that it is neither fully tested nor maintained. For more information on extracting accounting records from accounting data files, see “Reports and analysis” on page 28.

E-mail notification messages

Advanced Accounting sends e-mail messages informing you of activity related to data files.

E-mail notification must be manually configured to function. For more information about configuring e-mail notification, see “Configuring e-mail notification” on page 4.

The following messages are sent through e-mail notification.

Table 2. E-mail notification messages

Subject line	Body
`AACCT: File nearly full`	`1400-330: The accounting file is 90% full.`
`AACCT: File ready`	`1400-331: The accounting file is ready for processing.`

Table 2. E-mail notification messages (continued)

Subject line	Body
AACCT: Subsystem out of files	1400-332: The Advanced Accounting subsystem has run out of files for use.
AACCT: Subsystem out of kernel buffers	1400-333: The Advanced Accounting subsystem has run out of kernel buffers.
AACCT: File I/O error	1400-334: The accounting file encountered an I/O error while writing.

Most e-mail programs provide filtering capability so that shell scripts and commands can trigger specific actions when messages are received. Use the string AACCT to identify incoming messages related to Advanced Accounting. To identify the significance of the message, use the message number in the body of the message.

Configuring e-mail notification

You can configure e-mail notification for Advanced Accounting to send notifications to notify you when a data file is 90% full. Another notification can be sent when the Advanced Accounting subsystem has switched to a new data file.

In the following scenario, you will set up e-mail notification to the address ″anywhere@anywhere.com″. This scenario describes how to configure e-mail notification using either in SMIT or the command line. You can also use Web-based System Manager to configure e-mail notification.

Things to Consider

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Change to root user.
2. On a command line, type `smit admin_notify`.
3. With the cursor on the **Admin Notification Status** field, press the Tab key to change the e-mail notification status from Off to On.
4. In the **Email-ID for Notification** field, type `anywhere@anywhere.com`.
5. Press Enter to save your settings and turn on e-mail notification.

A valid e-mail address must first be specified using the SMIT interface, and then you can turn notification on and off on a command line. If e-mail notification is activated on a command line, the notifications will be sent to the most recently used e-mail address. To configure e-mail notification from the command line, type:

```
acctctl email anywhere@anywhere.com
acctctl email on
```

Projects

Projects represent billable entities such as users, departments, divisions, companies, or tasks. Each project is composed of a project number, project attribute, and project name, which collectively represent a project definition. Project definitions are entered into the project definition database.

Projects are written to accounting records. Report and analysis commands convert project numbers into project names by looking up entries in the system project definition database. Logically, *projects* are indices into critical business data (for example, customer name, billing address, account number, service level agreement) that are maintained by the billing application.

Project numbers are assigned through project assignment policies, which are composed of project assignment rules. Each rule contains classification criteria that when fully satisfied, yield a classification result. The *classification result* is a project list that is logically assigned to the object, usually a process, being classified. The classification criteria depends on the type of policy.

Project lists enable manual project assignment. If a list of projects is specified, users can change their current project assignment to another project in the list. This provides the capability to launch jobs under different projects, which is useful when you are performing work on behalf of multiple clients. The system administrator can assign multiple projects to any user, and then manually change the project assignment.

Project classification semantics

Project classification semantics are used to classify and assign projects.

For every exec(), initp(), setuid(), and setgid() subroutine, the process will be reclassified using the project assignment rules to determine if the current project assignment should be changed. If a project assignment rule is not loaded, or if a rule cannot be successfully applied, then the current project identifier is used.

The default project system identifier is zero (0). It is applied to the base system processes before accounting is enabled and it may be used to signify general system overhead. After it is assigned, a project is inherited from the parent process to the child process using the **fork()** kernel service and **creatp()** kernel service.

The use of the application filter varies between the **initp()** kernel service and the exec() subroutine. In the former, the command name of the kernel process that is being started is used to perform the classification. The command name is visible through the **ps** command. In the latter, classification is performed using the FID (inode + device number) of the executable, with the FID of the fully qualified path name that is specified by the policy. Only FIDs that can be computed at policy load time are accommodated.

If a process is classified through a rule that explicitly names an application, then that project identifier should be applied to the process and its children, because the intent is to label a block of work. This implies that subsequent exec(), setgid(), and setuid() subroutines in the process, as well as its children, do not perform project reclassification. Internally, these processes are identified with a sticky bit that indicates they were classified through a rule that explicitly named an application. The sticky bit is visible through the **ps -P** command.

The following is an example of the output after you run the **ps -P** command. The asterisk (*) preceding the *dev* project indicates it has a sticky bit attached to it:

```
UID    GID    PID    TTY    TIME    PROJECT    SUBPROJ    CMD
0      0      16922  pts/1  0:00    *dev       0          ps
0      0      19206  pts/1  0:00    *dev       0          acctras
0      0      22286  pts/1  0:00    *dev       0          ksh
```

When you load a new policy file, all of the processes in the system are reclassified, except those with a sticky bit. Processes with a sticky bit cannot be changed.

When a different project identifier is assigned to a process, a process accounting record is written to the **acct** file, so that the use of resources by a project may be accurately reported. Whenever this occurs, the accounting statistics for the process are reset to zero.

Manual project classification

Users can manually change their current project assignment.

For non-privileged users, authorization is provided through the policy file. The first project listed in the rule is considered to be the default project and is automatically chosen, unless the user changes his current

project assignment to another project in the list. Changing the current project assignment applies only to the current session and it is not silently applied to other jobs with the possible exception of the parent process (for example, the shell).

The following rule is an example of a project list.

Table 3. Example of a project list

User	Group	Application	Project(s)
User1	-	-	Chemistry, Biology

Because Chemistry is the first project in the list, it is the default project. User1 can change his current project assignment to Biology. The current project assignment is used as a parameter to project classification, which might include Biology as a potential project assignment.

Project lists can also be specified in rules that name applications, allowing administrators to associate projects with specific instances of applications. This is important for server processes, which may be externally configured to manage specific resources that have different cost structures.

This mechanism enables system administrators to charge indirectly for those resources that may not directly support usage-based accounting. For example, a system administrator that wants to charge different rates to access specific databases could start different instances of oracle to manage those databases.

The following rule an example of how applications are specified:

Table 4. Example of how applications are specified

User	Group	Application	Project(s)
-	-	/usr/bin/Oracle	Dataset1, Dataset2
User1	-	-	Chemistry, Biology, Dataset2

In the first rule, the application Oracle has two projects associated with it. If Oracle is started and the current project is not Dataset2, then Dataset1 is chosen. Otherwise, Dataset2 is chosen. If User1's current project assignment is Chemistry or Biology and he starts his database application, it will be assigned the project identifier Dataset1. User1 could also choose to change his current project assignment to Dataset2 and invoke a database application again, which would result in the project assignment of Dataset2.

For privileged users (root users or users with Advanced Accounting capability), there is a force option (the **-f** flag) that ignores the rules.

Project classification through environment variables

You can use environment variables to classify processes.

You can use the environment variables PROJECTNAME=*project name* and PROJECTID=*project id* to classify processes. If you have root authority or Advanced Accounting administrator capabilities, then the assignment is made without consulting project assignment rules. Otherwise, the project is validated against the loaded policy. If both PROJECTNAME and PROJECTID are specified, PROJECTNAME takes precedence over PROJECTID.

Relative project classification

You can enable project assignments to be made relative to user IDs and group IDs with Advanced Accounting.

To relatively classify a project, use the $UID and $GID keywords in a project list to tell the system that the project code must be computed. A simple expression in the form *keyword* or *keyword* + *constant* may be used, where *constant* is either a decimal or hexadecimal number (0xffff).

The following rule illustrates the use of relative project classification:

Table 5. Relative project classification

User	Group	Application	Project
*	-	-	$UID + 1000000

Disablement of accounting

You can disable accounting for selected processes through the classification process.

You can disable accounting by specifying a ″NoAccounting″ project list. This attribute is inherited from the parent process to child process.

The following example illustrates how this can be accomplished:

Table 6. Example of the NoAccounting specification in a project list

User	Group	Application	Project
Oracle	Oracle	/usr/*/oracle	NoAccounting
Root	Root	kbiod	NoAccounting

Oracle is classified in the first rule by the subroutine **exec()**. The NFS kproc in the second rule is classified by the subroutine **initp()**.

Project creation

You can create a project in Advanced Accounting.

Projects represent billable entities. Each user, department, division, or company that your business bills must be represented by a project in Advanced Accounting. The **projctl add** command is used to create a project. Once created, the project is stored in the project definition file, located by default in the **projdef** file in the **/etc/project** directory.

Note: When you are adding a project definition, make sure aggregation is disabled. To enable aggregation for a project definition, you must first create the project definition with aggregation off, and then manually enable aggregation for that project. For more information about data aggregation, see “Data aggregation” on page 27.

Creating a project definition

You can create a project definition that includes billable entities.

This procedure describes how to create a project definition. Projects represent billable entities. Each project is composed of a project number, project attribute, and project name, which collectively represent a project definition. The following scenario describes how to create a project definition from both SMIT and a command line. The following describes how to create a project definition called mktg.

Things to Consider

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

1. Change to root user.
2. From a command prompt, type: `smit add_proj`.
3. In the Project Name field, type: `mktg`.
4. Assign a project number to the project. For example, in the Project Number field, type `10`.
5. Verify that Aggregation field is set to **Off**. You cannot create a project in SMIT with the Aggregation field set to **On**.
6. Add an appropriate comments in the **Project Comments** field. For this scenario, in the Project Comments field, type `This is the default project for the Marketing department`.
7. Press Enter to save the values and create the project definition.

To create a project definition called **mktg** from the command line, type `projctl add mktg 10 off "This is the default project for the Marketing department"`.

Project commands and fast paths

You can manage your projects on the command line or using SMIT fast paths.

The following table shows the commands used to administer projects:

Table 7. Project commands

Task	Command	SMIT fast path
Add a project definition.	**projctl add** *projname projnumber* [*comment*] [{**-d** *projpath* | **-p** [*DN*]}]	**smit add_proj**
Load or reload a project definition.	**projctl ldprojs** [{**-g** [*DN*] | **-p** [*DN*]}] [**-d** *projpath*] [**-r**] [**-a**]	**smit load_proj**
Remove the specified project definition.	**projctl rm** *projname* [{**-d** *projpath* | **-p** [*DN*]}]	**smit rename_proj**
Show or change active project definitions.	**projctl chg** *projname* [**-p** *pid* [, *pid*]] [**-f**]	**smit show_chg_proj**
Merge two project definitions.	**projctl merge** *sourceprojpath* [**-d** *targetprojfile*]	None
Start a program with a project assignment.	**projctl exec** *projname* [*cmd line*] [**-f**]	**smit start_proj_prg**
Enable or disable aggregation for a project.	**projctl chattr agg** *projname* {**-s**|**-u**} [{**-d** *projpath* | **-p** [*DN*]}]	None
Show the policies that are loaded on the system.	**ojctl qpolicy** [**-g** [*DN*]]	
List all active project definitions.	**projctl qprojs** [**-n**]	None
List the specified project definition.	**projctl qproj** *projectname*	**smit show_chg_proj**
Change the project assignment for a process.	**projctl chg** *projname* [**-p** *pid* [, *pid*]] [**-f**]	**smit chg_proj_proc**
Show project assignments for a program.	**projctl qapp** *appname*	**smit show_proj_pgm**
Manage projects.	None	**smit work_project**
Remove project definitions.	None	**smit remove_admin_proj**

For more information, see *AIX Version 6.1 Commands Reference, Volume 4*.

Policies

Policies automate project assignment. A policy is comprised of classification criteria and classification results. The project assignments take place using subroutines and kernel services, such as **exec**, **initp**, **setuid**, and **setgid**.

You can use policies to classify data by user, group, application, or a combination of these attributes. Depending on the type of policy that is created, administrators can specify a user name, group name, application name, and project list in the policy file, although all four components do not have to be present for a policy to function.

Processes can be assigned in the following two ways:

- Using assignment rules when process classification attributes change. This is the most common way that processes are classified.
- Manually assigned to a class by a user with the required authority.

Admin policy

The admin policy uses the user name, group name, and application name process attributes to classify processes. The admin policy is application-based, and provides the ability to gather accounting statistics at the application level.

The admin policy supports the most filters and the use of Korn shell syntax wildcards. Configuring and maintaining Admin policies requires a standalone administration tool, such Web-based System Manager, SMIT, or an editor.

By default, the admin policy is located in the **/etc/project** directory. You can create alternate admin policies to use at various times. For example, you might want to have an admin policy that runs Monday through Friday, and another that runs on Saturday and Sunday. The alternate admin policies are stored in subdirectories of the**/etc/project/alter** root directory. You must specify a name for the subdirectory when you are creating a new admin policy.

Admin policy assignment rules

The admin policy consists of one or more assignment rules that are placed in the admin policy file.

Admin policy assignment rules must conform to the following syntax:

user name:*group name*:*application name*:*Project List*::*Optional comments go here*

For details on the values that are allowed in each field of the assignment rule, see the following table.

Table 8. User, group, and application rules

Type of rule	Description
user	Can contain either a hyphen (-) or at least one valid user name as defined in the **/etc/passwd** file. An exclamation point (!) can be used before a name to exclude a given user from the class. The list of user names is composed of one or more names, separated by a comma (,). Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax. If a nonvalid or an incorrect user name is used, Advanced Accounting ignores the entire rule. If a hyphen (-) is used, Advanced Accounting skips to the next field in the rule.
group	Can contain either a hyphen (-) or at least one valid group name as defined in the **/etc/passwd** file. An exclamation point (!) can be used before a name to exclude a given user from the class. The list of group names is composed of one or more names, separated by a comma (,). Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax. If an incorrect user name is used, Advanced Accounting ignores the entire rule. If a hyphen (-) is used, Advanced Accounting skips to the next field in the rule.

Table 8. User, group, and application rules (continued)

Type of rule	Description
application	Can contain either a hyphen (-), a list of application path names, or the command name of a kernel process. This is the path name of the applications (programs) run by processes included in the class. The application names will be either full path names or Korn shell patterns that match path names. The list of application names is composed of one or more path names, separated by a comma (,). An exclamation mark (!) can be used before a name to exclude a given application. At least one application in the list must be found at load time or the rule is ignored. Rules that are initially ignored for this reason might become effective later on if a file system is mounted that contains one or more applications in the list.

For process assignment policies, the classification criteria is the user name as listed in the **/etc/passwd** file, the group name as listed in the **/etc/groups** file, and fully qualified application name. The classification result is a project list. By default, the first project in the list is used, unless the user changes his current project assignment to a secondary project in the list.

Classification is done whenever an attribute value changes by comparing the value of these process attributes against lists of possible values given in the class assignment rules file, called *rules*. The comparison determines which rule is a match for the current value of the process attributes.

To classify the process, Advanced Accounting examines the top-level Admin policy for the active configuration to determine in which class the process belongs. For each rule in the file, Advanced Accounting checks the current values of the process attributes against the values and lists of values specified in the rule. Advanced Accounting goes through the rules in the order in which they appear in the Admin file, and classifies the process in the project corresponding to the first rule for which the process is a match. Therefore, the order of the rules in the rules file is significant.

The following is a list of the criteria used to determine whether the value of a process attribute matches the values of the same attribute field in the Admin policy file:

- If the field in the rules file has a value of hyphen (-), then any value of the corresponding process attribute is a match.
- If the value in one of the fields is preceded by an exclamation point (″!″), that value is always excluded.
- If the value in one of the fields is followed by an asterisk (*), then every match with that value will be recognized.

Examples of Admin policy rules

These examples show how you can use admin policy rules.

The admin policy allows you to specify more than one user, group, or application in their respective fields. For instance, if you wanted Frank, Bob, and John to be given the same attributes, you would specify the following syntax:

```
User1,User2,User3:-:-:Project List::Comments
```

The preceding syntax shows that User1, User2, and User3 will be treated the same way for this rule. The dashes in the group name and application fields are wildcards. You can also use an asterisk (’*’). As previously mentioned, you can also use wildcards to include all the values of a certain attribute. For example, if you wanted to include every user name beginning with B, you would type `B*` in the User Name field. Full Korn shell pattern matching syntax is allowed in all of the fields of the rule.

You can also set up your admin policy is to include certain users, but exclude others:

```
User1,!User2,User3:-:-:Project List::Comments
```

The preceding syntax shows that User1 and User3 will have the same attributes, but the policy will exclude User2.

Note:

1. The kernel only reads numeric values. In the above example, the user names User1, User2, and User3 are converted to numeric values after they are loaded into the kernel.
2. If any changes are made to the policy, it must be reloaded into the kernel using the **projctl** command.

Aliases for Admin policies

Aliasing provides a shorthand for referring to lists of users and groups, which simplifies admin policies, and makes them more readable and easier to maintain.

You can define an alias name for a list of users or groups so that you don't have to enter the full list of users, groups, or application names in a rule. A separate alias file is used with each admin policy and is placed in the same directory as its associated Admin policy.

To create an alias where User1, User2, and User 3 are grouped into an alias named `dev1` would be defined as follows:

```
dev1:User1,User2,User3::Development team 1
```

To use the alias name must be precede the alias name in the admin policy with a dollar sign ('$'). For example, using the `dev1` alias instead of User1, User2, and User3 in the admin policy looks similar to the following:

```
$dev1:-:-:Project List::This is a rule within a user alias
```

Multiple aliases, separated by a comma, can also be used. Aliases are excluded by placing an exclamation point (!) before the alias name in the admin policy rule.

Alternate Admin policies

You can create alternate Admin policies that are enforced in different situations that you define.

It is possible to create multiple Admin policies to reflect different billing strategies based on the time of the day or the day of the week. The **projctl** command is used to load alternate Admin policies. Previously loaded Admin policies do not have to be unloaded to load a new policy. The **cron** facility is used to load a policy at the a given time.

Alternate Admin policies are placed in the **/etc/project/alter** directory. For example, an alternate Admin policy with the name ″weekend″ is placed in the Admin file **/etc/project/alter/weekend/admin**. If this policy used aliases, they would be placed in the **/etc/project/alter/weekend/alias** alias file. SMIT and Web-based System Manager are used to create, change, show, load, unload, and remove Admin policies. Alternate policies are addressed in SMIT by changing the current focus to the name of the policy.

Creating an Admin policy

The Admin policy is used to automate project assignment.

This policy allows you to gather accounting statistics at the application level. The default Admin policy is named **admin** and is located in the **/etc/project** directory. If you create other admin policies, they are stored in a subdirectory of the **/etc/project/alter** directory. You can either create a subdirectory using the **mkdir** command, or using SMIT.

User and Group policies

The User and Group policies use the process attributes for users and groups. A user or group name and a project list constitute a project assignment rule within the User or Group policies. There is no file associated with a User or Group policy as user policy data is stored in a security database.

The User and Group policies provide project assignment based exclusively on user names or group names, depending on the policy. They are integrated with user administration. Project lists can be specified for users and groups when these users and groups are created.

The following table shows the structure of a user policy:

Table 9. Structure of a user policy

User name	Project list
User1	project1,project 2
User2	biology,chemistry

To create a Group policy, complete the following steps:

1. Create a group by typing the following:
   ```
   mkgroup staff
   ```
2. Create a project by typing the following:
   ```
   projctl add biology_dept 1200 Project Comment
   ```
3. Associate the project `biology_dept` with the newly created group `staff`:
   ```
   chgroup projects=biology_dept staff
   ```

Note: To create a user policy, complete Steps 1 through three, substituting **mkuser** and **chuser** for **mkgroup** and **chgroup**.

Once a user or group has been created, a project is associated with it. The users or groups can have several projects associated with them, but only one can be active at a time.

User or Group policy creation

You create User and Group policies by associating projects with a user or group.

User and Group policies are used to automate project assignments for users and groups. The **chuser** command and the **chgroup** command are used to associate projects with users and groups.

Policy commands and fast paths

You can manage your policies on the command line or using SMIT fast paths.

The following table shows the commands used to administer policies.

Table 10. Policy commands

Task	Command	SMIT fast path
Show the policies that are loaded on the system.	**projctl qpolicy**	**smit query_policy**
Load policies.	**projctl ldadm** [{**-g** [{[*name:*]*DN* | *name*}] | **-p** [{[*name:*]*DN* | *name*}] }] [**-d** *admproj*] [**-r**] [**-a**] **projctl ldusr** [**-a**] [**-r**] **projctl ldgrp** [**-a**] [**-r**] **projctl ld** [**-a**] [**-r**] **projctl ldall** [**-d** *localadm*] [**-r**] [**-a**]	**smit load_admin** **smit load_users** **smit load_groups**

Table 10. Policy commands (continued)

Task	Command	SMIT fast path
Unload policies.	**projctl** {**unldusr**|**unldgrp**|**unldall** | {{**unldrojs** | **unldadm**} [{**-p** [*name*] | **-g**}] [**-f**]}} [**-a**]	**smit unload_admin** **smit unload_users** **smit unload_groups**
Create an Admin policy.	None	**smit create_admin**
Show or change current focus policy.	None	**smit change_show_focus**
Remove an Admin policy.	None	**smit remove_admin**
Add a rule.	None	**smit add_admin_rule**
Remove a rule.	None	**smit remove_admin_rule**
Add a user alias.	None	**smit add_usr_alias**
Add a group alias.	None	**smit add_grp_alias**
Show or change the specified alias.	None	**smit chg_alias**
Remove the specified alias.	None	**smit remove_alias**
Create a project list for a user.	**chuser** *user*	**smit create_user**
Create a project list for a group.	**chgroup** *group*	**smit create_group**
Show or change a specified project list for a specified user.	**chuser projects=***projectlist user*	**smit change_show_user_list**
Show or change a project list for a group.	**chgroup** *group*	**smit change_show_group_list**
Remove a project list for a specified user.	**chuser projects=***user*	**smit remove_user**
Remove a project list for a specified group.	**chgroup projects=***group*	**smit remove_group**
Show project lists for all users.	**lsuser -a ALL**	None
Show project lists for all groups.	**lsgroup -a projects ALL**	None

For more information, see *AIX Version 6.1 Commands Reference, Volume 4*.

Application Resource Management interfaces

Applications use the Application Resource Management (ARM) interfaces to describe the transactional nature of their workloads.

Advanced Accounting supports ARM interfaces by recording information that is presented through these interfaces in the accounting data file. To use this information for charge-back purposes, it is necessary to understand the programming model associated with the ARM interfaces and the mechanism for passing critical business information to the Advanced Accounting subsystem, so that this information can be preserved for your billing application.

ARM interface structure

The ARM programming model has a fundamentally hierarchical structure.

Transaction instances are derived from transaction definitions that are defined when registering transactions. Application instances are derived from application definitions that are defined when registering applications. When starting a transaction, a transaction definition that should be used within an

application instance is specified, enabling the full set of information to be defined for each transaction instance. For example, all transactions have application and group names.

Advanced Accounting supports the ARM interfaces through a hierarchy of records:

- Application environment record
- Transaction environment record
- Transaction instance record

The application environment record describes a unique combination of application information including the application name, application group name, and properties (both identity and context). Each application environment record is assigned a unique identifier so that it can be referred to symbolically. This identifier is called the *application environment identifier*, and it is included within the transaction instance record.

The transaction environment record describes a unique combination of transaction information, including the transaction name and properties. Each transaction environment record is assigned a unique identifier so it can be referred to symbolically. This identifier is called the *transaction environment identifier*, and it is included with the transaction instance record.

Application environment identifiers and transaction environment identifiers are long lived, but they are not persistent in nature. Each time the system boots, the identifiers are regenerated. Advanced Accounting circumvents this problem by recording the application and transaction environment in each file so report and analysis commands should use the entry in the current file to determine the unique combination of values that should be applied to a transaction.

The transaction instance record is used to describe each individual transaction. It includes the application environment identifier and the transaction environment identifier defined above. These identifiers should be used to look up the corresponding application environment record and the transaction environment records so that the application names, application group names, and transaction names can be associated with the transaction instance. The intent is to minimize the amount of data that needs to be recorded with each transaction instancesince a lot of it is static in nature.

Advanced Accounting also supports the aggregation of ARM accounting data. For more information about data aggregation, see “Data aggregation” on page 27.

ARM interface programming model

The ARM interface programming model includes routines that perform basic functions.

The following table describes the basic programming model.

Table 11. ARM interface programming model

Interface	Description
arm_register_application	This routine registers the application identity with the ARM implementation. The application identity provides the fundamental instrumented application scope and basis for subsequent ARM 4.0 calls. This routine is called during application initialization.
arm_start_application	This routine establishes the started application instance with the ARM implementation, in preparation for making ARM 4.0 transaction calls. This interface is called during application initialization, after registering the application with ARM.

Table 11. ARM interface programming model (continued)

Interface	Description
arm_register_transaction	This routine registers a transaction identity with the ARM implementation. A transaction identity provides a category of transactions, commonly referred to as a transaction type, that will execute under the registered application for subsequent ARM 4.0 calls for transaction monitoring and measurement. This interface is called during application initialization after starting the ARM application instance.
arm_start_transaction	This routine establishes the started transaction instance with the ARM implementation. A started transaction instance provides the foundation for transaction monitoring and measurement. This routine is called during transaction processing.
arm_block_transaction	This routine indicates that the started transaction instance has become blocked behind a particular event. This routine is called during transaction processing.
arm_unblock_transaction	This routine indicates that the blocking event has been relieved for the started transaction instance. This routine is called during transaction processing.
arm_bind_transaction	This routine indicates that the current thread is performing on behalf of a started transaction instance. Binding enables the system to measure the processor utilization of a transaction because it establishes an exclusive processing relationship between the thread and the transaction. This routine is called during transaction processing.
arm_unbind_transaction	This routine indicates that the current thread is no longer performing on behalf of a started transaction instance. This routine is called during transaction processing.
arm_stop_transaction	This routine ends the started transaction instance recognized by the ARM implementation. A call to this interface is the expected way for an instrumented application to end a started transaction instance.
arm_stop_application	This routine ends the started application instance recognized by the ARM implementation. A call to this interface is the expected way for an instrumented application to end a started application instance.
arm_destroy_application	This routine deregisters the registered application. A call to this interface is the expected way for an instrumented application to deregister an application.

Parameters recognized by the ARM implementation

The ARM APIs provide a way to delineate application transactions through the use of a named set of parameters that are recognized by the ARM implementation and Advanced Accounting.

The parameters set for ARM enable the operating system to identify them and Advanced Accounting to measure and record them in the accounting data file. The ARM APIs also enable applications to describe their transactions, so that site-specific information can be provided. This is accomplished largely by convention.

Advanced Accounting recognizes the parameters shown in the following table.

Table 12. Parameters recognized by the ARM implementation and Advanced Accounting

Parameter	Description
application name	The application name is specified through the *app_name* parameter to the arm_register_application() interface. This parameter is set by the application and cannot be overridden by the user. It names the application, so a name similar to ″IBM® DB2 Universal Database™″ should be used.
application group name	The application group name is specified through the *app_group* parameter to the arm_start_application() interface. This parameter provides a grouping of the individual application instances that are collectively configured to provide an integrated service, so something identifiable such as ″Sample Supply Chain Management″ should be used.
transaction name	The transaction name is specified through the arm_register_transaction() interface. The name that is provided should be descriptive so the data file can be analyzed to determine the type of operation that transpired.
identity properties	Identity properties are used to specify properties that never change values. Identity properties can be specified for registered applications and transactions so they can be used at different levels to describe the fixed nature of an application or transaction. Identity properties can be used to identify discount rates or account codes that are known by the Advanced Accounting subsystem. Account codes (similar to project codes) should be specified through the predefined property: `EWLM: AIX: Account Class`
context properties	Context properties are used for information that changes. Context properties are associated with application and transaction instances, although Advanced Accounting does not capture accounting data for the latter. Only the name of the context property is captured for a transaction instance. Accounting codes (similar to project codes) can also be specified through context properties. In this cases, use the predefined property: `EWLM: AIX: Account Class`

Internally-generated transactional accounting data

Advanced Accounting also captures data that is generated internally. This information cannot be changed by the user, but is needed by accounting.

The following table describes the internally-generated data.

Table 13. Internally-generated transactional accounting data

Parameter	Description
User name and identifier	The user name and identifier is derived from the user sub-buffer that is specified when starting a transaction. It is intended to identify the agent that started the transaction. It can be an IP address or a machine name. It is not necessarily a UNIX user name or user ID.

Table 13. Internally-generated transactional accounting data (continued)

Parameter	Description
Response and queued times	Response and queued times describe the quality of service that was provided to the transaction. Response and queued times identify the elapsed time of the transaction and the time spent in the application before any real action was taken.
Resource utilization	The resource utilization of a transaction describes the physical use of resources such as the amount of processor time spent on the transaction. This statistic cannot always be computed because an application may work on multiple transactions simultaneously. In fact, this statistic will not be calculated, unless the application uses the arm_bind_thread() interface, which dedicates a thread to a single transaction.

Interval accounting

Interval accounting provides a way to collect accounting data at specified intervals. You can configure it to produce intermediate process records for active processes.

Records from interval accounting can be added to the completed process records to produce a more accurate bill reflecting the total use of the system by a user in a given time period. Interval accounting can also be configured to periodically capture accounting data for system resources like processors, memory, disks, network interfaces, and file systems. This information can be used to generate a bill that reflects the total use of a partition.

Usage information about system resources has other applications beyond chargeback. It can be used to perform capacity planning because it shows the use of physical resources like processors, memory, disks, and network interfaces. Interval accounting provides a time-based view of these resources, so that load can be determined, enabling capacity-based decisions to be made based on the data. For example, it is possible to determine the idle processor capacity in a given interval, which can be used to determine whether more processors are needed.

Interval accounting provides a historical view of resource use, which can be used for performance analysis. For example, the file system records can be examined to determine which file systems were busy when access to the system was slow. The data identifies multiple busy file systems served by the same disk adapter. This information can be used to balance file systems over disk adapters. You do not always know which file sets are being accessed at a given time, so having the ability to replay a log with this information is an asset.

System interval accounting

System interval accounting collects system-related information.

System interval accounting collects resource use information about system resources such as processors, disks, network adapters, file systems, and memory usage. This information is used to profile the use of your system. For most purposes, an interval of once per hour should be sufficient.

The system interval does not collect process accounting data.

Process interval

Process interval accounting collects process-related information.

The process interval is used to capture information about long-running jobs such as data modeling applications that run for months. The standard process completion record is sufficient for most processes. Therefore, the process interval should be relatively large, so that it does not produce unnecessary data. The process interval writes a record for every active process, including newly started jobs. In some situations this can amount to a large number of records, so the process interval should be set to capture information once per day (1 440 minutes). However, if process records are being aggregated automatically by the system, then set the process interval for once per hour.

Interval accounting commands and fast paths

You can manage interval accounting using the command line or SMIT fast paths.

By default, interval accounting is turned off. The following table shows the commands you can use to turn on and turn off system-level and process-level interval accounting:

Table 14. Interval accounting commands

Task	Command	SMIT fast path
Enable system interval accounting, specified in number of minutes by the *time* parameter, or turn off system interval accounting.	**acctctl isystem** {*time*|off }	**smit system_interval**
Enable process interval accounting, specified in number of minutes by the *time* parameter, or turn off process interval accounting.	**acctctl iprocess** {*time*|off}	**smit process_interval**
Query the state of Advanced Accounting.	**acctctl**	None

For more information, see *AIX Version 6.1 Commands Reference, Volume 1*.

Hosted accounting policies

An accounting policy is hosted by another system enabling a single point of control to be established for the management of accounting policies.

Using a single point of control simplifies the administration and maintenance of accounting policies. You can use this capability to account for a user who logs into multiple machines across an enterprise. Alternatively, or in conjunction with this capability, you can define server-specific billing strategies. The Advanced Accounting subsystem enables multiple policies to be defined concurrently for a server. You can use this flexibility to implement the right solution for your business.

You implement hosted accounting policies using the Lightweight Directory Access Protocol (LDAP), which defines a standard mechanism for accessing and updating information in a directory (a database) either locally or remotely using a client-server model. The Advanced Accounting subsystem uses this technology to provide a single point of control for the management and distribution of accounting policies and projects to LDAP clients. You must configure each LDAP client system to access accounting data on the LDAP server, but once configured, the client behaves as if the policy were defined locally on that system.

The Advanced Accounting subsystem is fully integrated with user authentication. When creating a new user or group, you can optionally specify a project list for that user or group. This project list is part of the definition of that user or group and is automatically delivered to the client system when the user logs in. The project list is just another user attribute.

In general, you should establish project definitions with the same scope as the user. If a user is defined globally through LDAP, then the user's project definitions should also be defined globally through LDAP.

Similarly, if a user is defined locally on a particular system, then the user's project definitions should be defined locally on that system. This simplifies the billing strategy, but is not required by the Advanced Accounting subsystem. Both project repositories can be loaded at the same time to accommodate local and LDAP users. It is recommended that Local and LDAP project definitions are defined in a non-overlapping way, although the Advanced Accounting subsystem does not enforce this. Projects are resolved based on the order that the project repositories are loaded. Local projects may be given precedence over LDAP projects by loading the local project repository before the LDAP project repository, and vice versa. It is also possible to only use local or LDAP projects.

The following advanced-accounting data sets can be used with LDAP:

- Project definitions
- Project lists for LDAP users and groups (for example, LDAP User and Group Policies)
- Admin policy

Admin policies can also be stored on an LDAP server. The Admin policy provides an alternative mechanism for performing classification that in some ways is easier to manage than specifying a project list for each user. The Admin policy is a collection of assignment rules that can be more easily updated because it is not distributed across many individual user definitions. The Admin policy is managed as a single entity. You can define both local- and LDAP-based Admin policies.

You can enable the following policies at the same time. The order of evaluation is:

1. Local Admin policy
2. LDAP Admin policy
3. User accounting policy
4. Group accounting policy

The LDAP server is not Advanced Accounting-aware and can be maintained at a different software level. You can upload a schema that describes Advanced Accounting data to the server, so that it can be used to distribute accounting policies and project definitions without any special knowledge of project and policies.

You use the **mkprojldap** command to configure the connection between the LDAP server and client. Specifically, you use this command to upload the LDAP schema associated with the Advanced Accounting data to the server. This command is also used to define the location on the server where the data is to be stored for a particular client. This enables you to implement a different Admin policy and project repository for each client system, if desired. These items are individually configurable to provide the maximum flexibility. One reason for implementing a different Admin policy could be for administrative purposes, such as making the policy reflect the set of authorized users. Or, you might need to accommodate a different billing strategy for a particular server. For example, the use of server X is always charged to account Y, or server X is intended to be used for projects W and Z only.

You must individually configure each client system for accounting purposes. You must specify the set of policies that should be activated on each system by using the **projctl** command. This command has been extended to provide new functions such as uploading and downloading LDAP-based projects and Admin policies. In general, once the client system has been set up, the location of the policy and project repository is transparent to the end user.

The Advanced Accounting subsystem produces accounting data locally. The **acctctl** command must still be used to define data files and to manage them on an ongoing basis. However, you might want to place these files on shared storage subsystems such as a Storage Area Network (SAN) or in a distributed file system like NFS or General Parallel File System™ (GPFS™) so that a billing application can have access to all of the data.

You can use the **ps** command to show the origin of a project that has been assigned to a process. This information is also recorded in the accounting record for a process, so that report and analysis tools can properly match the assigned project to the proper project repository, assuming that the tool is aware of multiple project repositories. This situation is best avoided by defining non-overlapping ranges of projects for local and LDAP-based projects.

Multisystem accounting policy files

You can upload or download a project definition file and Admin policy files to an LDAP server.

The multisystem accounting policy includes the following data sets:

- Default LDAP projects
- Default LDAP Admin policy
- Alternate LDAP Admin policy

These data sets are stored on the local system in the following directories:

- **/etc/project/ldap/projdef**
- **/etc/project/ldap/admin**
- **/etc/project/ldap/alter/policy name/admin**

You can load multiple Admin policies to the LDAP server. You can use this capability to implement time-based policies, such as policies for peak and off-peak usage.

Configuring an LDAP server to host accounting policies

You must configure the LDAP server to host accounting policies before they can be provided to a client system.

You can perform the setup procedure from any LDAP client that is configured in a general way to access the LDAP server. It is not necessary to perform the setup procedure on the LDAP server. To configure an LDAP server to host accounting policies, you must first upload the Advanced Accounting subsystem schema, which is shipped with AIX. The schema describes the layout of accounting data so that the LDAP server does not need to be Advanced Accounting-aware.

There is no requirement that the LDAP server be at the same software level as the client.

To set up the LDAP server, run the following command for each LDAP server:

```
mkprojldap –u –h hostname -D bindDN -w BindPassword
```

You must then decide where to store accounting data on the LDAP server. Each client system asks for accounting data at a particular location, so it is important to understand the layout of accounting data on the server. You must understand the billing strategy to define the proper layout on the server. If you want to deploy a server-specific billing policy, then you should use Admin policies, since they can be targeted at a specific machine. In this case, you should place the Admin policy and project definitions in a location on the LDAP server that is reserved for that system.

If you want to use an enterprise-level policy that always classifies a user in the same way, then you should use the User or Group policies. In this case, you must define the project repository in a global location on the LDAP server so that it can be accessed by each client. Other strategies are also possible.

To define a base location on the server where you can store accounting data, use the following command:

```
mkprojldap –s –h hostname -D bindDN -w BindPassword -i InstallPoint
```

For example:

```
mkprojlap –s –h ldap.svr.com –D cn=root –w passwd –i -p cn=aixdata,o=ibm -a cn=aixdata,o=ibm
```

This enables Admin policies and project definitions to be stored on the server below the install point. You must run this command as a root user once for each base location.

Configuring an LDAP server to host accounting policies using SMIT

You can configure an LDAP server to host multisystem accounting policies using SMIT.

To configure an LDAP server to host multisystem accounting policies using SMIT, use the following procedure.

1. Log in as a root user.
2. Access the SMIT menu for the LDAP server setup using the following SMIT path: **smitty aacct** → **Manage Advanced Accounting Subsystem** → **Manage LDAP configuration** → **LDAP server setup**.
3. Enter the server name, Bind DN, Bind password, and the install points for projects and policies in the relevant fields.
4. Press Enter to configure the LDAP server for the Advanced Accounting subsystem.

Configuring an LDAP client for accounting using the command line

You can set up an LDAP client for accounting purposes using the command line.

These instructions assume that the client system is configured as an LDAP client.

Use the **mksecldap** command to establish the basic connection between the LDAP client and server.

Use the **mkprojldap** command to provide accounting-specific parameters to establish an accounting-aware LDAP client.

Use the **projctl** command to configure projects and policies as required by the billing strategy. The final step involves specifying a refresh policy for data provided by an LDAP server.

To configure an LDAP client, perform the following steps:

1. Log in as a root user.
2. Run the **mkprojldap –c –D** *bindDN* -w *bindPWD* -a *default-adminDN* –p *default-projectDN* command, where *default-adminDN* and *default-projectDN* are the base locations on the LDAP server on which the client will look for accounting data. This command adds accounting-specific information to the LDAP configuration file (**ldap.cfg**) and restarts the LDAP client daemon. This is an example of the command: **mkprojldap –c -D cn=testroot -w testpwd –a ou=adminpolicy,ou=aacct,cn=aixdata –p ou=projects,ou=aacct,cn=aixdata**.
3. If you want to upload LDAP projects or Admin policies to the LDAP server, you can do so at this point.
4. If you want to configure the current system to automatically use LDAP projects when a policy is loaded, run the **projctl ldprojs -g -a** command. Projects are resolved in the order that they are loaded. Therefore, if you want local projects to take precedence, run the **projctl ldprojs –a** command first. The **–g** flag indicates that data is to be retrieved from the LDAP server. You must configure both sources if you intend to use both sources.
5. If you want to configure the current system to automatically load an LDAP Admin policy, when accounting is started, run the **projctl ldadm –g -a** command. You might also want to configure a local Admin policy. You can do this by running **projctl ldadm –a**. Unlike projects, there is not a precedence issue between Admin policies. The local Admin policy takes precedence over the LDAP Admin policy.
6. Use the **cron** facility to periodically refresh projects and Admin policies that are loaded from the LDAP server. The interval might be once an hour or once a day depending on the site-specific policy for accommodating new users.

You can also perform the above steps through SMIT.

After the client has been configured to use the accounting capabilities provided by an LDAP server, it is not necessary to have LDAP-specific knowledge to administer the Advanced Accounting subsystem, unless you want to add new project definitions to the LDAP project repository or modify an LDAP-based accounting policy or project definition.

Project definitions on an LDAP server

You can upload the project definition to an LDAP server using a command line or SMIT.

By default, the project definitions are uploaded from the default LDAP project definition path **/etc/project/ldap/projdef**.

Uploading the project definition to an LDAP server using the command line

After you have configured the LDAP server to host accounting data, you can upload the project definitions to the configured DN in the LDAP server from the command line.

By default, the project definitions are uploaded from the default LDAP project definition path **/etc/project/ldap/projdef**. You can specify alternate path names using the **–d** flag with the **projctl** command.

To upload the project definition file to an LDAP server from the command line, perform the following steps:

1. Place the file containing the project definitions in the directory **/etc/project/ldap/** with name **projdef**. The directory **/etc/project/ldap/** acts as local repository for the LDAP projects and policies.
2. Run the **projctl ldprojs –p –d /etc/project/ldap/** command. This command uploads the **projdef** file to the default project DN on the LDAP server. To upload **projdef** to a non-default DN, specify the DN parameter with **-p**.

Uploading the project definition to an LDAP server using SMIT

When the LDAP server has been configured to host accounting data, you can upload the project definitions to the configured DN in the LDAP server using SMIT.

To upload the project definition file to an LDAP server through SMIT, use the following procedure:

1. Log in as a root user.
2. Access the SMIT menu **Upload Project Definitions to LDAP Server** using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Upload Project Definitions to LDAP Server**.
3. Enter the local project definition file path and destination location on the LDAP server.
4. Press Enter to upload the project definition file to the LDAP server.

Admin policies on an LDAP server

You can upload the Admin policy to an LDAP server.

You can use the **projctl ldadm –p** command to upload an Admin policy to the LDAP server. By default, the source Admin policy is found at the file location **/etc/project/ldap/admin**.

To specify that a different policy be uploaded, use the **–d** flag. By default, the target location for the policy on the LDAP server is defined by the Admin policy DN that was established when the client was set up. You can upload the policy to a different target location on the LDAP server by specifying a DN with the **-p** flag. The DN is an optional parameter.

If the Admin policy contains new project definitions, or if this is the first time that the policy is uploaded to the server and the LDAP project definitions have not been uploaded, then you must upload the project definitions so that the policy is resolved correctly when it is downloaded to another client. The Advanced Accounting subsystem downloads project definitions automatically from the LDAP server when a policy is

loaded, but it does not automatically upload projects when a policy is uploaded. You must perform this step, because the project repository on the server is a critical resource and the upload operation is a replacement operation.

Uploading the Admin policy to an LDAP server using the command line

You can upload the Admin policy to an LDAP server from the command line.

To upload the Admin policy file to an LDAP server, use the following procedure.

1. Place the file containing the Admin policies in the directory **/etc/project/ldap/** with name **admin**. The directory **/etc/project/ldap/** acts as local repository for the LDAP projects and policies.
2. As a root user, run the **projctl ldadm –p –d /etc/project/ldap/** command. This command uploads the Admin policy file to the default project DN on the LDAP server. To upload the file to a non-default DN, specify the DN parameter with **-p**.

Uploading the Admin policy to an LDAP server using SMIT

You can upload the Admin policy to an LDAP server using SMIT.

To upload the Admin policy file to an LDAP server through SMIT, use the following procedure:

1. Log in as a root user.
2. Access the SMIT menu Upload Admin Policy to LDAP Server using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Automatic Project Assignment** → **Work with Admin Policies** → **Upload Admin Policy to LDAP Server**.
3. Enter the local Admin policy file path and destination location on the LDAP server. By default, the policy file will be uploaded with name **default**. A different name can be specified using the **Admin Policy Name** option.
4. Press Enter to upload the Admin policy to the LDAP server.

Projects and policies in the kernel

Before you use the LDAP projects and policies, you must first load the LDAP projects and policies into the kernel. You can load projects and policies in the kernel using a command line or SMIT.

Loading LDAP projects into the kernel using the command line

Before you use the LDAP projects and policies, you must first load the LDAP projects and policies into the kernel. You can load LDAP projects into the kernel using the command line.

When loading the LDAP projects into the kernel, the projects should be present on the default project's DN on the LDAP server.

To load the LDAP projects into the kernel, perform the following steps:

1. Log in as a root user.
2. Run the **projctl ldprojs –g** command. This command downloads the **projdef** file from the LDAP server to the **/etc/project/ldap/** directory and then loads the project definitions in the file into the kernel.

Loading LDAP projects into the kernel using SMIT

If you want to use the LDAP projects and policies, you must first load the LDAP projects and policies into the kernel. You can load LDAP projects into the kernel using SMIT.

When loading the LDAP projects into the kernel, the projects should be present on the default project's DN on the LDAP server.

To load the LDAP projects into the kernel through SMIT, perform the following steps:

1. Log in as a root user.

2. Access the SMIT menu **Load/Re-load Project Definitions** using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Load/Re-load Project Definitions**.
3. Select LDAP as the project repository.
4. Press Enter to load the LDAP project definitions into the kernel.

Loading LDAP Admin policies into the kernel using the command line

You can load the LDAP Admin policies into the kernel using the **projctl ldadm –g** command.

To load the LDAP projects into the kernel, the Admin policy file must be present on the default administrator DN on the LDAP server. To load LDAP projects into the kernel, perform the following steps:

1. Log in as a root user.
2. Run the **projctl ldprojs –g** command. This command downloads the **projdef** file from the LDAP server to the **/etc/project/ldap/** directory, and then loads the project definitions in the file into the kernel.

Loading LDAP Admin policies into the kernel using SMIT

You can load the LDAP Admin policies into the kernel using the SMIT.

To load the LDAP Admin policies into the kernel through SMIT, use the following procedure.

1. Log in as a root user.
2. Access the SMIT menu **Load/Re-load Admin Policy** using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Automatic Project Assignment** → **Work with Admin Policies** → **Load/Re-load Admin Policy**.
3. Select LDAP as the **Current focus is on**.
4. Press Enter to load the LDAP Admin policies into the kernel.

Projects and policies in a local file

You can download projects and policies to a local file using a command line or SMIT.

Downloading LDAP project definitions to a local file using the command line

You can download project definitions from the LDAP server to the client to append new project definitions or to delete obsolete ones, and then upload the file. You can edit the project definition file directly, rather than using the **projctl** command to perform these tasks.

You can download an LDAP project definition file to a local directory using the **projctl** command. This command downloads the project definition file from the default project DN on the LDAP server to the specified local directory. To download the project definition file from a specific Distinguished Name (DN), you must specify the DN parameter with the **–g** flag.

The Advanced Accounting subsystem downloads project definitions automatically from the LDAP server when an accounting policy is loaded (if the client system has been configured to use LDAP as a project repository). The project definitions are downloaded during the client setup phase, when you explicitly load project definitions from an LDAP server and indicate that they should be automatically configured. To do this, use the **projctl ldprojs -g -a** command.

LDAP projects are cached on each client system. You must define a refresh policy for periodically downloading accounting data from the server. There is a SMIT menu item that allows you to specify the frequency of policy refreshes. This delay might impact the ability of the client to properly classify a new user, if the new user is given access to the system before the project is automatically downloaded to that client system. In this event, the user can use the system, but the user is not properly classified. A **SYSLOG** message indicating that the user is not properly classified is generated automatically. You can manually invoke the refresh command on the client systems where users are expected to perform their work. The refresh command is **projctl ldall -r**.

Downloading LDAP project definitions to a local file using SMIT

You can download project definitions from the LDAP server to the client to append new project definitions or to delete obsolete ones, and then upload the file. You may prefer to edit the project definition file directly, instead of using the **projctl** command to perform these tasks.

To download LDAP project definitions through SMIT, use the following procedure.

1. Log in as a root user.
2. Access the SMIT menu **Load/Re-load Project Definitions** using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Download Project Definitions from LDAP server**.
3. Enter the local project definition file path and location on the LDAP server.
4. Press Enter to download the project definition file.

Downloading LDAP Admin policies to a local file using the command line

You can download a local copy of the LDAP Admin policies to the client using the command line. You can make any required updates to the local copy of the LDAP Admin policies file before uploading.

To download the LDAP Admin policy file to a local directory, use the **projctl ldadm –g –d local-dir** command. This command downloads the Admin policy file along with the associated alias and project definition files from the default administrator and project DNs on the LDAP server to the specified local directory. To download the Admin policy file from a specific DN, specify the DN parameter with the **–g** flag. When the Admin policy is downloaded from a specific DN, the project definition file will not be downloaded.

Downloading LDAP Admin policies to a local file using SMIT

You can download a local copy of the LDAP Admin policies to the client using SMIT. You can make any required updates to the local copy of the LDAP Admin policies file before uploading.

To download the LDAP Admin policy file through SMIT, perform the following steps:

1. Log in as a root user.
2. Access the SMIT menu Load/Re-load Admin Policy using the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Automatic Project Assignment** → **Work with Admin Policies** → **Download Admin Policy from LDAP Server**.
3. Enter the local Admin policy file path and location in the LDAP server fields. If no Admin policy file name is specified, the default policy file will be downloaded. To download a named Admin policy file, fill in the **Admin Policy Name** field.
4. Press Enter to download the Admin policy.

Updates to LDAP projects

You can perform any required updates to LDAP projects without downloading them to the client machine.

To update LDAP projects, use the **projctl** subcommands **add**, **rm** and **chattr**.

To update the project repository on the LDAP server, use the **-p** flag.

Updates to LDAP projects using the command line

You can add or remove a new project definitions to the LDAP project definition file from the command line.

To add a new project definition to the LDAP project definition file, use the **projctl add projname projnumber –p** command. This adds the new project definition to the project definition file on LDAP default project DN.

To remove a project definition from the LDAP project definition file, use the **projctl rm projname –p** command. This removes the specified project definition from the project definition file on the default project DN.

To enable the aggregation property of a project definition in an LDAP project definition file, use the **projctl chattr agg projname –s –p** command. This enables the aggregation property of the project definition specified on the LDAP project definition file.

For all of the above operations, if a project definition on a specific LDAP DN needs to be modified, then specify the DN parameter with the **–p** flag.

Updates to LDAP projects using SMIT

You can add or remove a new project definitions to the LDAP project definition file using SMIT.

To add a new project definition to the LDAP project definition file through SMIT path, use the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Add Project Definition**.

To remove a project definition from the LDAP project definition file through SMIT, use the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Remove Project Definitions**.

To enable the aggregation property of a project definition in an LDAP project definition file using SMIT, use the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Show/Change Project Definitions**.

For all of the above operations, if a project definition on a specific LDAP DN needs to be modified, then specify the DN parameter with the **–p** flag.

Removal and unload of projects and policies

You can unload and remove projects and policies using a command line or SMIT.

Unloading LDAP projects from the kernel

You can unload LDAP projects from the kernel.

To unload the LDAP projects from the kernel, use the **projctl unldprojs –g** command. This command removes only the project definitions that are loaded from the LDAP repository.

To unload the LDAP projects from the kernel through SMIT, use the following SMIT path:**smitty aacct** → **Manage Project Definitions and Assignments** → **Project Definitions** → **Unload Active Project Definitions**.

Unloading LDAP policies from the kernel

You can unload the LDAP Admin policies from the kernel.

To unload the LDAP Admin policy from the kernel, use the **projctl unldadm –g** command. This command only removes the Admin policies loaded from LDAP repository.

To unload the LDAP Admin policy from the kernel through SMIT, use the following SMIT path: **smitty aacct** → **Manage Project Definitions and Assignments** → **Automatic Project Assignment** → **Work with Admin Policies** → **Unload Admin Policy**.

Removing LDAP projects from the server

You can remove LDAP projects from the server.

To remove the LDAP project repository from the LDAP server, use the **projctl unldprojs –p** command. You can also use this command to remove projects that are not located under the default DN that is configured for the client system, by specifying the default DN as an argument to the **-p** flag.

Removing LDAP Admin policies from the server

You can remove LDAP Admin policies from the server.

To remove an LDAP Admin policy from the LDAP server, use the **projctl unldadm –p** command. This command removes the default Admin policy from the default DN that is configured for the client. To remove an alternate Admin policy, you must specify the name of the policy to be removed with the **-p** flag. You can also use this command to remove Admin policies that are not located under the default DN by specifying that DN explicitly. This can be combined with an alternate policy name in the following manner: **-p** *name*:*DN*. The name and the DN must be separated by a colon. Colons are not allowed in policy names or DNs.

Data aggregation

Data aggregation is a method of accumulating data into an accounting record that is otherwise presented through multiple records.

Aggregated data is written periodically, according to the intervals described in “Interval accounting” on page 17.

Note:

1. Interval accounting must be enabled in order to use data aggregation.
2. Set the process interval and system interval to 60 minutes.

Accounting data is totaled inside the kernel with no impact to applications or middleware. The data is made available by interval accounting, which is a kernel function that periodically writes these records to the **acct** file. When the kernel aggregates records, it maps them to a set of aggregated records that are managed internally. These records are pending in the sense that they have been input to the system, but have not been committed to the recording mechanism inside Advanced Accounting. Interval accounting is used to commit the aggregated records so that they are written to the **acct** file.

Because aggregated data is recorded using different data structures, you must verify that the billing application recognizes these structures. Consult the documentation provided with your billing application to determine if your billing application supports data aggregation. Data aggregation can be enabled or disabled at the system level or project level.

System-level data aggregation commands and fast paths

You can manage system-level data aggregation using the command line or SMIT fast paths.

System-level data aggregation commands allow you to turn on and off process data aggregation, kernel extension data aggregation, and ARM transaction data aggregation. The following table lists the commands you can use to turn on and off process data aggregation, kernel extension data aggregation, and ARM transaction data aggregation.

Table 15. System-level data aggregation commands

Task	Command	SMIT fast path
Enable or disable system-wide process aggregation.	**acctctl agproc** {**on**|**off**}	**smit system_paggr**
Enable or disable system-wide aggregation for third-party kernel extensions.	**acctctl agke** {**on**|**off**}	**smit system_kaggr**

Table 15. System-level data aggregation commands (continued)

Task	Command	SMIT fast path
Enable or disable system-wide aggregation for ARM transactions.	**acctctl agarm** {**on**|**off**}	**smit system_aaggr**
Query the overall accounting state	**acctctl**	None

For more information, see *AIX Version 6.1 Commands Reference, Volume 1*.

Project-level data aggregation commands

You can manage project-level data aggregation using the command line or SMIT fast paths.

The following table shows the commands to turn on and off aggregation for individual projects.

Table 16. Project-level data aggregation commands

Task	Command	SMIT fast path
Enable or disable aggregation for a project, specified by the *projname* parameter.	**projctl chattr agg** *projname* {**-sl-u**} [**-d** *projpath*]	None
Query the aggregation state for all projects.	**projctl qprojs** [**-n**]	None
Query the aggregation state for the specified project.	**projctl qproj** [*projectname*]	None

For more information, see *AIX Version 6.1 Commands Reference, Volume 4*.

Reports and analysis

The Advanced Accounting subsystem provides accounting data for a wide variety of resources that are typically included within charge-back mechanisms.

The format of the accounting data file, as well as the format of the individual accounting records are in the **sys/aacct.h** are in the header file. Use the **acctrpt** command to parse the accounting data file and produce accounting reports. You can use this command to produce three types of accounting reports: Process Accounting reports, LPAR Accounting reports, and Transaction Accounting reports, and Workload Partition Accounting reports.

Along with the **acctrpt** command, AIX provides a sample program that parses the accounting data file. This program is **readaacct**. This program can be used to analyze accounting data and import accounting data into a spreadsheet. This command has the following syntax:

```
/usr/samples/aacct/readaacct [-F file] [-t trid] [-b begin_time] [-e end_time] [-c] [-h] [-@ <wpar>]
```

This command has the following flags:

- The **-c** flag is used to display information in colon-separated format.
- The **-h** flag is used to display information about the file, such as the hostname and machine model, and serial number of where the data was produced.
- The **-b** and **-e** flags give a time-based view of the information.
- The **-t** flag gives a record-based view of the information.
- The **-@** flag gives records specific to a Workload Partition.

The following is an example of the output displayed after running the **readaacct -F /tmp/afile -h** command:

```
# readaacct -F /tmp/afile -h
File Name=/tmp/a
Version=0
Flags=0
Offset=3084288
File Size=3145728
State=2
ID=1
First Time=1087266596
Last Time=1087301336
System ID=IBM,01025990A
System Model=IBM,7040-681
Host Name=bigboylp9
Partition Name=bigboylp9
Partition Number=9
------------------------------------------------------------
Transaction ID=1
Flags=f1
Transaction Project=0
Sub project ID=0
Transaction start time=6-14-2004 21:29:56
UID=0
GID=0
PID=335912
eWLM Service Class=0
Flags=1
Command Name=acctctl
Controlling Terminal's Device Number=23,5
Process Start Time=1087266596
WLM Class key=7770295601810996315
Incrementing Statistics:
Elapsed process time=0.032406 seconds
Elapsed thread time=0.032406 seconds
Process CPU time=0.015238 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=5 seconds
Elapsed Page seconds of virtual memory=5 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
```

Examples of the Process Accounting report

You can generate Process Accounting reports.

Process Accounting report

The following is an example of the Process Accounting report:

```
 # /usr/bin/acctrpt –f /var/aacct/acctdata
                                                                                (C) PELAPSE   TELAPS   CPU   (sec)
                                                                                (M) VMEM      DMEM     PMEM  (pg)
                                                                                (F) LFILE     DFILE          (MB)
TIMESTAMP  PROJID   UID      GID      PID      CMD          STARTED  EXITED (S) LSOCKET   DSOCKET        (MB)
---------  ------   ---      ---      ---      ---          -------  ------ --- -------   -------  ------
11151936   System   root     system   524386   acctctl      11151936 E       C: 0.3824    0.3824   0.2394
                                                                               M: 25        27       0
                                                                               F: 0.00      0.00
                                                                               S: 0.00      0.00
11151936   System   root     system   524388   nfssync_kpro 11151936 E       C: 0.0008    0.0008   0.0003
```

```
                                                    M: 0        0        0
                                                    F: 0.00     0.00
                                                    S: 0.00     0.00
```

Aggregated Process Accounting report

The following is an example of the Aggregated Process Accounting report. The flags **–P**, **-U**, **-G**, and **–C** of the **acctrpt** command facilitate preparing aggregated process accounting reports. These flags correspond respectively to project ID, user ID, group ID, and command name. The administrator can control the order in which the data is presented by changing the order of the flags. The following is an example of an Aggregated Process Accounting report based on all user IDs:

```
 # /usr/bin/acctrpt –U ALL –f /var/aacct/acctdata
                                              (C) PELAPSE  TELAPSE  CPU      (sec)
                                              (M) VMEM     DMEM     PMEM     (pg)
                                              (F) LFILE    DFILE             (MB)
PROJID   UID      GID      CMD          CNT    (S) LSOCKET  RSOCKET           (MB)
         ---                             ---    --- -------- -------- --------
-        root     -        -            590    C:  54880.8  281458.2 60.3
                                               M:  6        14100294 14098216
                                               F:  32.3     1.4
                                               S:  0.0      19.8
-        104      -        -            28     C:  3.3      3.3      1.0
                                               M:  186      299      708
                                               F:  8.3      0.0
                                               S:  0.0      0.0
```

Example of the Logical Partition Accounting report

You can generate a Logical Partition (LPAR) Accounting report.

LPAR Accounting Report

The following is an example of the LPAR Accounting report. You use the **–L** flag of the **acctrpt** command to generate accounting reports to specific LPAR resources. The following is an example of file system statistics:

```
# /usr/bin/acctrpt –L FILESYS –f /var/aacct/acctdata
File Systems Accounting Report
------------------------------
CNT    DEVNAME            MOUNTPT    FSTYPE RDWR    OPEN    CREATE  LOCKS   XFERS(MBs)
7      specfs             specfs     16     48853   1066    0       0       1.2
7      pipefs             pipefs     16     3827    0       0       0       3.3
7      /export/sp1n1fs2u /farm/cwu   18     4917    4510    750     0       28.0
```

Example of the Transaction Accounting report

You can generate a Transaction Accounting report.

Transaction Accounting report

The following is an example of the Transaction Accounting report. To generate accounting reports for ARM transactions, use the **–T** flag with the **acctrpt** command.

```
# /usr/bin/acctrpt –T –f /var/aacct/acctdata
                    (A)  CLASS          NAME            USER     GROUP    TRANSACTION
PROJID   CNT        (T)  RESPONSE       QUEUED          CPU                            (sec)
------   ---        ---  --------       -------         ----    -------   ------------
System   144        A:  -              WebSphere        -       server1   URI
                    T:  0.00           0.00           0.00
System   32         A:  -              IBM Webserving   -       IBM_SERV  Apache/1.3.28(Unix)
                    T:  0.00           0.00           67.01
```

Accounting records

There are multiple types of records produced by Advanced Accounting.

Accounting records produced by Advanced Accounting are defined in the **sys/aacct.h** file. The following table describes these records.

Table 17. Accounting records

Accounting record	Description
Pad record (type 0)	This record does not provide any meaningful accounting data. Report and analysis tools should skip this record. It is generated for alignment purposes only.
Process record (type 1)	This record is written when a process exits, when a process is reclassified (**setUser ID()**, **chproj()**, **exec()**) and when the system is reclassified. This record is written by the process interval. This record contains the following information: • User ID • Group ID • Process ID • Process flags (exited, core, killed by signal, killed by checkpoint) • Base command name • WLM class • Controlling terminal • Process start time (in seconds from the EPOCH) • Process elapsed time in micro seconds • Combined thread elapsed time in micro seconds • Process (combined threads) processor time in micro-seconds • Elapsed page seconds of disk pages • Elapsed page seconds of real pages • Elapsed page seconds of virtual memory • Local logical file I/O (JFS, J2) in bytes • Other logical file I/O (NFS, DFS™) in bytes • Local socket I/O (UNIX domain and loopback) in bytes • Remote socket I/O in bytes The process start time and Process ID can be used to correlate interval records for a particular process. The exit flag can be used to distinguish between interval and exit records.

Table 17. Accounting records (continued)

Accounting record	Description
Aggregated process record (type 2)	This record is derived from the process record. A different record is produced for each user by project. This record is produced by the process interval and contains the following information: • User ID • Time of first record aggregated (in seconds from the EPOCH) • Number of processes aggregated • Aggregate process elapsed time in micro seconds • Aggregate thread elapsed time in micro seconds • Aggregate process (combined threads) processor time in micro seconds • Aggregate elapsed page seconds of disk pages • Aggregate elapsed page seconds of real pages • Aggregate elapsed page seconds of virtual memory • Aggregate local logical file I/O (JFS, J2) in bytes • Aggregate other logical file I/O (NFS, DFS) in bytes • Aggregate local socket I/O (UNIX domain and loopback) in bytes • Aggregate remote socket I/O in bytes

Table 17. Accounting records (continued)

Accounting record	Description
Aggregated application record (type 3)	This record is derived from the process record. Records are produced at the user, project, and application level. This record is similar to the aggregated process record, except that the application is named. This record is produced when the process is classified with an application specific rule, which is supported only through the Admin policy. This record is produced by the process interval and contains the following information: • User ID • Time of first record aggregated (in seconds from the EPOCH) • Inode of the command that generated the project classification • Device number of the command that generated the project classification • Number of applications aggregated • Aggregate process elapsed time in micro seconds • Aggregate thread elapsed time in micro seconds • Aggregate process (combined threads) processor time in micro seconds • Aggregate elapsed page seconds of disk pages • Aggregate elapsed page seconds of real pages • Aggregate elapsed page seconds of virtual memory • Aggregate local logical file I/O (JFS, J2) in bytes • Aggregate other logical file I/O (NFS, DFS) in bytes • Aggregate local socket I/O (UNIX-domain and loopback) in bytes • Aggregate remote socket I/O in bytes

Table 17. Accounting records (continued)

Accounting record	Description
Processor and memory use record (type 4)	This record provides information about the use of processors and when the size of the large page pool changes. This record is also generated during pre-migration and post-migration and by the system interval. This record contains the following information: • Reason the record was generated • Number of online logical processors • Entitled physical processor capacity of the partition • Total idle time, in milliseconds • Total I/O wait time, in milliseconds • Total kernel process time, in milliseconds • Total user process time, in milliseconds • Total interrupt time, in milliseconds • Size of physical memory, in megabytes • Size of the large page pool, in megabytes • Large pages in use, in megabytes • Number of page ins from paging space • Number of page outs to paging space • Number of start I/Os • Number of page steals • Elapsed time since start of interval, in milliseconds
Policy record (type 5)	This record is written when a policy file is loaded or unloaded. It is provided for informational purposes only. This record contains the following information: • Type of policy: Admin, User, or Group • Load or unload
File system activity record (type 6)	This record describes the use of file systems at the system level. A separate record is generated for each mounted file system. This record is produced by the system interval and has the following information: • Total bytes transferred through read and write • Total number of read and write requests • Total number opens • Total number of creates • Total number of locks • File system type • Device name • Mount point

Table 17. Accounting records (continued)

Accounting record	Description
Network interface I/O record (type 7)	This record provides information about the use of network interfaces at the system level. This record is produced by the system interval and contains the following information: • Logical name of the network interface • Number of I/Os • Total bytes transferred
Disk I/O record (type 8)	This record provides information about the use of disks at the system level. A separate record is written for each logical disk device. This record is produced by the system interval and contains the following information: • Logical name of the disk • Total disk transfers • Total reads • Total writes • Block size of the disk transfer
Lost data record (type 9)	This record provides information about accounting records that were deleted because Advanced Accounting did not have the ability to record them. This occurs when all of the accounting data files are full. When the ability to write new accounting records is restored, Advanced Accounting produces the lost data record describing the outage. This record contains the following information: • Number of lost records • Number of microseconds of lost processor time associated with process records • The time that data loss began (in microseconds from EPOCH) • Number of microseconds of lost processor time associated with third party kernel extension records
Server VIO record (type 10)	This record is produced in hosting partitions. A separate record is produced for each logical device that is shared with a client partition. The system interval can be used to periodically produce this record. This record contains the following information: • Client partition number • Server unit ID • Device logical unit ID (LUN) • Number of bytes in • Number of bytes out

Table 17. Accounting records (continued)

Accounting record	Description
Client VIO record (type 11)	This record is produced in client partitions. It describes the use of virtual devices in client partitions. A separate record is recorded for each instance of a virtual device. The system interval may be used to periodically produce this record. This record contains the following information: • Server partition number • Server unit ID • Device logical unit ID • Number of bytes in • Number of bytes out
Third-party kernel extension common aggregation record (type 12)	This record provides accounting information for the named accounting record. It is derived from aggregated accounting records that are produced by third-party kernel extensions. This record is written to Advanced Accounting by the system interval. This record contains the following information: • Command name of the kernel extension (from u-block) • Third-party kernel extension transaction ID, in the range of 129 to 256 • Number of accounting records that have been aggregated • Resource use, or accumulated processor time, for this class of transactions • Time of first record aggregated (in seconds from the EPOCH)
ARM application environment record (type 13)	This record describes an application environment instance. It is created from data that is passed to the operating system through the **arm_register_application()** system call and the **arm_start_application()** system call. This record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: • Character set in which the data in this record is recorded • Application environment identifier • Offset to application name • Offset to application group • Offset to application identity properties • Offset to application context properties The operating system attempts to record the content of the application environment in each accounting data file, so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate the dependency between accounting data files.

Table 17. Accounting records (continued)

Accounting record	Description
ARM transaction environment record (type 14)	This record describes a transaction environment instance. It is created from data that is passed to the operating system through the **arm_register_transaction()** system call. This record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: • Character set in which the data in this record in recorded • Transaction environment identifier • Offset to transaction name • Offset to application identity properties • Offset to application context properties (names only) The operating system attempts to record the content of the transaction environment in each accounting data file (not guaranteed), so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate the dependency between accounting data files.
ARM transaction instance record (type 15)	This record describes an ARM transaction instance. It is created from data that is passed to the operating system through the **arm_start_transaction()** and the **arm_stop_transaction()** system calls. It is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: • Completion status of the transaction • Application environment identifier • Transaction environment identifier • Offset to user identifier (not User ID) • Offset to user name (not uname) • Offset to accounting code • Response time, in milliseconds • Queued time, in milliseconds • Resource use The application and transaction environment identifiers are defined respectively in the application and transaction environment records. These records must be used to associate application names, application groups, transaction names, and properties with the transaction instance.

Table 17. Accounting records (continued)

Accounting record	Description
ARM aggregated transaction instance record (type 16)	This record is produced instead of the ARM transaction instance record (type 15), when aggregation is enabled for ARM transactions. This record contains the following information: • Completion status of the transaction • Time of first record aggregated (in seconds from EPOCH) • Application environment identifier • Transaction environment identifier • Offset to user identifier (not user ID) • Offset to user name (not uname) • Offset to accounting code • Aggregate response time, in milliseconds • Aggregate queued time, in milliseconds • Aggregate resource use
Project definition record (type 17)	This record provides a list of project definitions. It is written when the project definition file is loaded. Multiple records may be needed to record all project definitions. This record is used to provide the full set of project information in each data file, so that data files may be treated as stand-alone entities. This may not be required by the billing application, depending on the nature of the billing application. This feature may be disabled by disabling the project definition accounting record. This record is variable in length and contains the following information: • Number of projects • Number of bytes in the project definition area • Project definition area

Table 17. Accounting records (continued)

Accounting record	Description
WPAR process record (type 33)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record is written when a process in an Application WPAR exits, when a process is reclassified (**setUser ID()**, **chproj()**, **exec()**) and when the system is reclassified. This record is written by the process interval. This record contains the following information: • User ID • Group ID • Process ID • Process flags (exited, core, killed by signal, killed by checkpoint) • Base command name • WLM class • Controlling terminal • Process start time (in seconds from the EPOCH) • Process elapsed time in micro seconds • Combined thread elapsed time in micro seconds • Process (combined threads) processor time in micro seconds • Elapsed page seconds of disk pages • Elapsed page seconds of real pages • Elapsed page seconds of virtual memory • Local logical file I/O (JFS, J2) in bytes • Other logical file I/O (NFS, DFS) in bytes • Local socket I/O (UNIX domain and loopback) in bytes • Remote socket I/O in bytes • WPAR Name The process start time and Process ID can be used to correlate interval records for a particular process. The exit flag can be used to distinguish between interval and exit records.

Table 17. Accounting records (continued)

Accounting record	Description
WPAR aggregated process record (type 34)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record is derived from the WPAR process record of an Application WPAR. A different record is produced for each user by project. This record is produced by the process interval and contains the following information: • User ID • Time of first record aggregated (in seconds from the EPOCH) • Number of processes aggregated • Aggregate process elapsed time in micro seconds • Aggregate thread elapsed time in micro seconds • Aggregate process (combined threads) processor time in micro seconds • Aggregate elapsed page seconds of disk pages • Aggregate elapsed page seconds of real pages • Aggregate elapsed page seconds of virtual memory • Aggregate local logical file I/O (JFS, J2) in bytes • Aggregate other logical file I/O (NFS, DFS) in bytes • Aggregate local socket I/O (UNIX domain and loopback) in bytes • Aggregate remote socket I/O in bytes • WPAR Name

Table 17. Accounting records (continued)

Accounting record	Description
WPAR aggregated application record (type 35)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record is derived from the WPAR process record of an Application WPAR. Records are produced at the user, project, and application level. This record is similar to the aggregated process record, except that the application is named. This record is produced when the process is classified with an application specific rule, which is supported only through the Admin policy. This record is produced by the process interval and contains the following information: • User ID • Time of first record aggregated (in seconds from the EPOCH) • Inode of the command that generated the project classification • Device number of the command that generated the project classification • Number of applications aggregated • Aggregate process elapsed time in micro seconds • Aggregate thread elapsed time in micro seconds • Aggregate process (combined threads) processor time in micro seconds • Aggregate elapsed page seconds of disk pages • Aggregate elapsed page seconds of real pages • Aggregate elapsed page seconds of virtual memory • Aggregate local logical file I/O (JFS, J2) in bytes • Aggregate other logical file I/O (NFS, DFS) in bytes • Aggregate local socket I/O (UNIX-domain and loopback) in bytes • Aggregate remote socket I/O in bytes • WPAR Name

Table 17. Accounting records (continued)

Accounting record	Description
WPAR processor and memory use record (type 36)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record provides information about the use of processors by System/Application WPAR and when the size of the large page pool changes. This record is also generated during pre-migration and post-migration and by the system interval. This record contains the following information: • Reason the record was generated • Number of online logical processors • Entitled physical processor capacity of the partition • Total idle time, in milliseconds • Total I/O wait time, in milliseconds • Total kernel process time, in milliseconds • Total user process time, in milliseconds • Total interrupt time, in milliseconds • Size of physical memory, in megabytes • Size of the large page pool, in megabytes • Large pages in use, in megabytes • Number of page ins from paging space • Number of page outs to paging space • Number of start I/Os • Number of page steals • Elapsed time since start of interval, in milliseconds • WPAR Name
WPAR file system activity record (type 38)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record describes the use of file systems specific to a System/Application WPAR at the system level. A separate record is generated for each mounted file system. This record is produced by the system interval and has the following information: • Total bytes transferred through read and write • Total number of read and write requests • Total number opens • Total number of creates • Total number of locks • File system type • Device name • Mount point • WPAR Name

Table 17. Accounting records (continued)

Accounting record	Description
WPAR network interface I/O record (type 39)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record provides information about the use of network interfaces by a system/Application WPAR at the system level. This record is produced by the system interval and contains the following information: • Logical name of the network interface • Number of I/Os • Total bytes transferred • WPAR Name
WPAR third-party kernel extension common aggregation record (type 44)	This record is produced only when WPAR accounting is enabled on Global WPAR. This record provides accounting information for the named accounting record. It is derived from aggregated accounting records of an Application WPAR that are produced by third-party kernel extensions. This record is written to Advanced Accounting by the system interval. This record contains the following information: • Command name of the kernel extension (from u-block) • Third-party kernel extension transaction ID, in the range of 129 to 256 • Number of accounting records that have been aggregated • Resource use, or accumulated processor time, for this class of transactions • Time of first record aggregated (in seconds from the EPOCH) • WPAR Name

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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.

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
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
DB2 Universal Database
DFS
General Parallel File System
GPFS
IBM

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Readers' Comments — We'd Like to Hear from You

AIX Version 6.1
Advanced Accounting Subsystem

Publication No. SC23-6606-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: aix6koub@austin.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC23-6606-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 04XA-905-6C006
11501 Burnet Road
Austin, TX 78758-3493

Fold and Tape | **Please do not staple** | Fold and Tape

SC23-6606-00

Cut or Fold
Along Line

Printed in U.S.A.

SC23-6606-00