

IEEE Standard for Information Technology — Portable Operating System Interface (POSIX™) — Part 2: Shell and Utilities — Amendment 1: Batch Environment

Sponsor
**Portable Applications Standards Committee
of the
IEEE Computer Society**

Approved December 13, 1994
IEEE Standards Board

Abstract: This amendment is part of the POSIX series of standards for applications and user interfaces to open systems. It is a supplement to IEEE Std 1003.2-1992, which defines the applications interface to a shell command language and a set of utility programs for complex data manipulation. This supplement defines the user interface to a distributed batch queuing environment and includes application environments.

Keywords: batch, batch extensions, distributed batch, queuing

POSIX is a trademark of the Institute of Electrical and Electronics Engineers, Inc.

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright ©1995 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1995.
Printed in the United States of America

ISBN 1-55937-512-4

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

foo

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contents

	PAGE
Introduction	iv
Section 1: General	1
1.1 Scope	1
1.3 Conformance	2
Section 2: Terminology and General Requirements	5
2.1 Conventions	5
2.2 Definitions	5
Section 8: Batch Environment Option Services and Utilities	11
8.1 General Concepts	11
8.2 Batch Services	15
8.3 Common Behavior for Batch Environment Option Utilities	35
8.4 qalter – Alter batch job	38
8.5 qdel – Delete batch jobs	48
8.6 qhold – Hold batch jobs	50
8.7 qmove – Move batch jobs	54
8.8 qmsg – Send message to batch jobs	56
8.9 qrerun – Rerun batch jobs	59
8.10 qrls – Release batch jobs	62
8.11 qselect – Select batch jobs	65
8.12 qsig – Signal batch jobs	74
8.13 qstat – Show status of batch jobs	77
8.14 qsub – Submit a script	82
Annex B (normative) C-Language Bindings Option	95
B.2 C Numeric Limits	95
Annex D (informative) Bibliography	97
Annex E (informative) Rationale and Notes	99
E.11 Batch Environment Option Services and Utilities	103
Annex F (informative) Portability Considerations	123
F.1 User Requirements	123
F.2 Portability Capabilities	123
F.3 Profiling Considerations	124
Annex I (informative) External Dependencies	125
I.1 Introduction	125
I.2 System Services	125
I.3 Networking and Communications	125

I.4	Resource Management		126
I.5	DOD Security		131
I.6	Host and User Validation		131
I.7	Data Transfer		131
I.8	Accounting		131
I.9	Unified System Administration		132
I.10	User Notification		132
Alphabetic Topical Index			133

TABLES

Table 8-1	– Environment Variable Summary		16
Table 8-2	– Next State Table		18
Table 8-3	– Results/Output Table		19
Table 8-4	– Batch Services Summary		27
Table 8-5	– Environment Variable Values (Utilities)		83
Table E-1	– Symbolic Constants		107
Table E-2	– Reserved Resource Names		109
Table E-3	– Option-Argument Matrix		112
Table E-4	– Command-Option Matrix		113
Table E-5	– Command-Operand Matrix		113
Table E-6	– Attribute Matrix		114
Table I-1	– Required Reserved Resource Names		127
Table I-2	– Recognized Reserved Resource Names		127

Introduction

(This introduction is not a normative part of IEEE Std 1003.2d-1994, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX™) — Part 2: Shell and Utilities — Amendment 1: Batch Environment, but is included for information only.)

The purpose of this amendment is to define a standard batch and general queuing extension to “POSIX.2” (the base standard). This definition is based on the Net-work Queuing System (NQS), which is in use among many large UNIX[®] system sites. This is intended for systems implementors and applications software developers.

Within this document, the term “POSIX.2d” refers to this amendment itself.

Organization of the Amendment

This amendment is divided into three normative sections, one normative annex, and five informative annexes:

- Statement of scope and list of conforming implementation options (Section 1)
- Definitions and abbreviations (Section 2)
- Batch processing concepts, batch server capabilities and behavior, and the user interface (Section 8)
- C-Language Bindings Option (Annex B)
- Bibliography (Annex D)
- Rationale and Notes (Annex E)
- Portability Considerations (Annex F)
- External Dependencies (Annex I)

Related Standards Activities

Activities to extend the POSIX family of standards to address additional requirements are in progress, and similar efforts can be anticipated in the future.

The following areas are under active consideration at this time or are expected to become active in the near future:¹⁾

- (1) Language-independent service descriptions of POSIX.1 {8}
- (2) C, Ada, and FORTRAN language bindings to (1)
- (3) Verification testing methods
- (4) Realtime facilities
- (5) Secure/trusted system considerations
- (6) Network interface facilities
- (7) System administration
- (8) Graphical user interfaces
- (9) Profiles describing application- or user-specific combinations of open systems standards for supercomputing, multiprocessor, and batch extensions; transaction processing; realtime systems; and multiuser systems based on historical models
- (10) An overall guide to POSIX-based or related open systems standards and profiles

Extensions are approved as “amendments” or “revisions” to this document, following the IEEE and ISO/IEC Procedures.

Approved amendments are published separately until the full document is reprinted and such amendments are incorporated in their proper positions.

If you have an interest in participating in the PASC working groups addressing these issues, please send your name, address, and phone number to the Secretary, IEEE Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA, and ask to have it forwarded to the chair of the appropriate PASC working group. If you have an interest in participating in this work at the international level, contact your ISO/IEC national body.

1) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue, NW, Washington, DC 20036-1903, USA; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

IEEE Std 1003.2d-1994 was prepared by the P1003.15 working group and sponsored by the Portable Applications Standards Committee and Operational Environments of the IEEE Computer Society. At the time this amendment was approved, the membership of the P1003.15 working group was as follows:

Portable Applications Standards Committee

Chair: Lowell Johnson
 Vice-Chairs: Andy Bihain
 Andrew Josey
 Lorraine Kevra
 Barry Needham
 Charles Severance
 Jon Spencer
 Treasurer: Pete Smith
 Secretary: Charles Severance

P1003.15 Working Group Officials

Chair: Norm Aaronson
 Karen Sheaffer (1987-1993)
 Vice-Chair: Norm Aaronson
 Dave Evans (1989-1993)
 Stuart McKaig (1988-1989)
 Charisse Castagnoli (1987-1988)
 Dave Evans (1987)
 Editor: Jim Tanner
 Secretary: Bill Mar (1990-1993)
 John Caywood (1988-1990)

Working Group

The following persons, who have been members of the P1003.15 Working Group at one time or another, have actively participated in the development of this amendment:

Norm Aaronson	Dave Evans	Judy Richards
Richard Alexander	Sig Handelman	Ed Russell
Jason Behm	Robert Henderson	Richard Russell
Jonathan Brown	Jan Jakubowski	Karen Sheaffer
Terrie Carver	Charles Johnson	Charles Smith
Charisse Castagnoli	Ted Johnson	Jim Tanner
John Caywood	J. Ted Kline	David Teater
Myles Connors	Bill Laurune	Mike Thomas
Laurie Costello	Bernard Lint	Dave Tweten
Fred Crowner	William Mar	Bob Venable
Amy Daul	Stuart McKaig	Dave Wallace
John Davis	A. Ray Miller	Howard Walter
Chuck DeBaun	Ray Miller	Grace Wiechman
Richard Edwards	Robert H. Mueller	Dave Wright
	Dean Nelson	

96 The following persons were members of the P1003.2d balloting group that
 97 approved the amendment for submission to the IEEE Standards Board:

98	Richard Alexander	<i>SHARE Organizational Representative</i>	
99	Robert Boucher	<i>Uniform Organizational Representative</i>	
100	Hal Jespersen	<i>Technical Editor of IEEE Std 1003.2-1992</i>	
101	Roland McGrath	<i>Free Software Foundation Organizational Representative</i>	
102	Nick Stoughton	<i>EurOpen Organizational Representative</i>	
103	Norman Aaronson	James Hegerty	Judith Richards
104	Timothy Baker	Robert L. Henderson	John Roman
105	Jason Behm	Jan Jakubowski	Patric Savage
106	E. Loren Buhle	Judy S. Kerner	Fritz Schulz
107	William Cox	Joseph King	Karen L. Sheaffer
108	Fred D. Crowner	Steve Kiser	Andrew N. Sheppard
109	John F. Davis	Robert Knighten	Richard Stallman
110	Charles DeBaun	Thomas M. Kurihara	Krystyne Supplee
111	Dave Decot	Will Leslie	James G. Tanner
112	Steven Deller	Donald Lewine	Ravi Tavakley
113	Richard L. Edwards	William Mar	David E. Tweten
114	David G. Evans	Pete Meier	Mark-Rene Uchida
115	Michel Gien	Bruce Potter	Paul Wanish
116	Mark D. Guzzi	Joseph Ramus	Matthew Wicks
117	Sigmund W. Handelman	Ray Ricco	Oren Yuen
118	Barry Hedquist		George R. Zerdian

When the IEEE Standards Board approved this amendment on December 13, 1994, it had the following membership:

Wallace S. Read, Chair **Donald C. Loughry, Vice Chair**

Andrew G. Salem, Secretary

Gilles A. Baril	Donald N. Heirman	Joseph L. Koepfinger*
Bruce B. Barrow	Richard J. Hollenman	D. N. "Jim" Logothetis
José A. Berrios de la Paz	Jim Isaak	L. Bruce McClung
Clyde R. Camp	Ben C. Johnson	Marco W. Migliaro
James Constantino	Sonny Kasturi	Mary Lou Padgett
Stephen L. Diamond	Lorraine C. Kevra	Arthur K. Reilly
Donald C. Fleckenstein	E. G. "Al" Kiener	Ronald H. Reimer
Jay Forster*	Ivor N. Knight	Gary S. Robinson
Ramiro Garcia		Leonard L. Tripp

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
Robert E. Hebner

Paula M. Kelty
IEEE Standards Project Editor

POSIX is a trademark of the Institute of Electrical and Electronics Engineers, Inc.

UNIX[®] is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

IEEE Standard for Information Technology — Portable Operating System Interface (POSIX™) — Part 2: Shell and Utilities — Amendment 1: Batch Environment

Section 1: General

1.1 Scope

⇒ **1.1 Scope.** *Add the following text in the proper hierarchical sequence:*

1.1.1 Scope of the Batch Environment Option

When the Batch Environment Option is included, the amended standard has additional scope. The list of utilities and features is extended to provide users a common method to submit work for deferred processing. This specification represents an amendment for distributed batch environments. The amendment is intended to support application and user portability and system interoperability in heterogeneous environments. Distributed batch environments include traditional batch processing and other generic queue processes.

NOTE: The Batch Environment Option will be developed as a series of amendments. Each amendment will describe a part of the batch queuing environment and the user of program interface to that portion. The parts of the batch queuing environment are batch servers, user utilities, administrator utilities, library interface, and data stream encoding.

The first of the Batch Environment Option amendments provides a definition of application environments and a user utility interface, and it creates a framework for an extensible network protocol.

This amendment is comprised of the following major components:

- (1) Terminology, concepts, definitions, and specifications that govern structures, headers, environment variables, and related requirements
- (2) Description of application environments
- (3) Description of batch services
- (4) Description of user utility interfaces
- (5) Rationale and external requirements

The following services are outside the scope of this amendment:

- Security
- Directory services
- File transfer
- User notification
- Network administration
- User name mapping
- Authentication

The following issues will be addressed in the future:

- Description of network protocol including data stream encoding
- Description of system administrative utility interface
- Programmatic interface
- Resource control
- Networked resource management

1.3 Conformance

⇒ **1.3.1.3 Conforming Implementation Options.** *Add the following symbolic constants to the list in alphabetic order:*

{POSIX2_PBS} The system supports the Batch Environment Option (see Section 8).

NOTE: The Batch Environment Option is a combination of mandatory and optional services and utilities. The {POSIX_PBS} symbolic constant implies the system supports all the mandatory services and utilities.

{POSIX2_PBS_ACCOUNTING}

The batch server supports the accounting capability described in this amendment (see 8.2.2.1.1).

50 {POSIX2_PBS_CHECKPOINT}
51 The batch server supports the checkpoint/restart capability
52 described in this amendment (see 8.2.2.1.2).

53 {POSIX2_PBS_LOCATE}
54 The batch server supports the Locate Job Request
55 described in this amendment (see 8.2.3.5).

56 {POSIX2_PBS_MESSAGE}
57 The batch server supports the Job Message Request
58 described in this amendment (see 8.2.3.3).

59 {POSIX2_PBS_TRACK}
60 The batch server supports the Track Job Request described
61 in this amendment (see 8.2.3.16).

1.3 Conformance

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05, 2026 at 04:23:20 UTC from IEEE Xplore. Restrictions apply.

Section 2: Terminology and General Requirements

2.1 Conventions

⇒ **2.1.1 Editorial Conventions.** Add the following table entry to Table 2-1 in alphabetic order:

Reference	Example
Batch Environment Option Attribute	<i>Account_Name</i>

2.2 Definitions

⇒ **2.2.2 General Terms.** Add the following definitions in alphabetic order (disregarding the subclause numbers shown here):

2.2.2.195 access list: A list of user IDs and group IDs of those users and groups authorized to place jobs in a queue.

An access list is associated with a queue. A batch server uses the access list of a queue as one of the criteria in deciding to put a job in a queue.

2.2.2.196 authentication: The process of validating a user or process to verify that the user or process is not a counterfeit.

2.2.2.197 authorization: The process of verifying that a user or process has permission to use a resource in the manner requested.

To assure security, the user or process would also need to be authenticated before granting access.

2.2.2.198 batch administrator: A person who is authorized to use all restricted batch services.

2.2.2.199 batch client [client]: A computational entity that utilizes batch services by making requests of batch servers.

Batch clients often provide the means by which users access batch services, although a batch server may act as a batch client by virtue of making requests of another batch server.

In this amendment, the term *client*, unless otherwise qualified, denotes a *batch client*.

2.2.2.200 batch job [job]: A set of computational tasks for a computing system.

Batch jobs are managed by batch servers.

Once created, a batch job may be executing or pending execution. A batch job that is executing has an associated session leader (a process) that initiates and monitors the computational tasks of the job.

In this amendment, the term *job*, unless otherwise qualified, denotes a *batch job*.

2.2.2.201 batch job attribute [job attribute]: A named data type whose value affects the processing of a batch job.

The values of the attributes of a batch job affect the processing of that job by the batch server that manages the job.

The attributes defined for a batch job are called the batch job attributes.

In this amendment, the term *job attribute*, unless otherwise qualified, denotes *batch job attribute*.

2.2.2.202 batch node [node]: A host containing part or all of a batch system.

A *batch node* is a host meeting at least one of the following conditions:

- Is capable of executing a batch client
- Contains a routing queue
- Contains an execution queue

In this amendment, the term *node*, unless otherwise qualified, denotes a *batch node*.

2.2.2.203 batch operator [operator]: A person who is authorized to use some, but not all, restricted batch services.

In this amendment, the term *operator*, unless otherwise qualified, denotes a *batch operator*.

2.2.2.204 batch queue [queue]: A manageable object that represents a set of batch jobs and is managed by a single batch server.

NOTE: Each batch job managed by a batch server is a member of a single batch queue managed by that server.

Such a set of batch jobs is called a queue largely for historical reasons. Jobs are selected from the queue for execution based on attributes such as priority, resource requirements, and hold conditions.

Two types of batch queues are described in this amendment: routing queues and execution queues.

In this amendment, the term *queue*, unless otherwise qualified, denotes a *batch queue*.

2.2.2.205 batch queue attribute [queue attribute]: A named data type whose value affects the processing of all jobs that are members of the queue.

A batch queue has attributes that affect the processing of jobs that are members of the queue.

The attributes defined for a batch queue are called the batch queue attributes.

In this amendment, the term *queue attribute*, unless otherwise qualified, denotes *batch queue attribute*.

2.2.2.206 batch server [server]: A computational entity that provides batch services.

In this amendment, the term *server*, unless otherwise qualified, denotes a *batch server*.

2.2.2.207 batch server name: A string that identifies a specific server in a network.

A string of characters in the portable character set (see 2.4) used to specify a particular server in a network.

2.2.2.208 batch service: Computational and organizational services performed by a batch system on behalf of batch jobs.

Batch services are of two types: *requested* and *deferred*. See Table 8-4.

2.2.2.209 batch system [system]: A collection of one or more batch servers.

In this amendment, the term *system*, unless otherwise qualified, denotes a *batch system*.

2.2.2.210 batch user: A person who is authorized to make use of batch services.

2.2.2.211 deferred batch service: A service that is performed as a result of events that are asynchronous with respect to requests.

NOTE: Once a batch job has been created, it is subject to deferred services.

2.2.2.212 destination: The batch server in a batch system to which a batch job should be sent for processing.

Acceptance of a job at a destination is the responsibility of a receiving batch server. A destination may consist of a batch server specific portion, a network wide portion, or both. The batch server specific portion is referred to as the queue. The network wide portion is referred to as a batch server name.

2.2.2.213 destination identifier: A string that identifies a specific destination.

A string of characters in the portable character set (see 2.4) used to specify a particular destination.

2.2.2.214 directive: A line from a file that is interpreted by the batch server.

The line is usually in the form of a comment and is an additional means of passing options to the **qsub** utility. See 8.14.7.2.

2.2.2.215 job identifier: A unique name for a job.

A name that is unique among all other job identifiers in a batch system and that identifies the server to which the job was originally submitted.

2.2.2.216 job name: A label that is an attribute of a job.

The job name is not necessarily unique.

2.2.2.217 job owner: The *username@hostname* of the user submitting the job, where *username* is a user name defined by Section 2.2.2.88 of POSIX.1 {8} and *hostname* is a *network host name* (2.2.2.223).

2.2.2.218 job priority: An attribute used in selecting a job for execution.

A value specified by the user that may be used by an implementation to determine the order in which jobs will be selected to be executed. *Job priority* has a numeric value in the range -1024 to 1023 .

NOTE: The *job priority* is not the execution priority (*nice value*) of the job.

2.2.2.219 job state: An attribute of a batch job.

The state of a job determines the types of requests that the batch server that manages the job can accept for the job. Valid states include **QUEUED**, **RUNNING**, **HELD**, **WAITING**, **EXITING**, and **TRANSITING**.

2.2.2.220 name service: A service that assigns names that are unique within the name space, and that can translate a unique name into the location of the named entity.

2.2.2.221 name space: The environment within which a name is known to be unique.

2.2.2.222 network: A collection of interconnected hosts.

The term *network* in this document is used to refer to the network of hosts. The batch system is used to refer to the network of batch servers.

2.2.2.223 network host name: A string of characters in the portable character set (see 2.4) that identifies a specific host in a network.

2.2.2.224 network pathname: The pathname of a file that includes the name of the host on which it resides.

This amendment allows two syntaxes for a network pathname:

pathname@hostname
hostname:pathname

2.2.2.225 queue position: The place a job occupies in a queue.

This place is relative to other jobs in the queue and defined in part by submission time and its priority (see *job priority* 2.2.2.218).

2.2.2.226 queue priority: The maximum job priority allowed for any job in a given queue.

The *queue priority* is set and may be changed by users with appropriate privilege. The priority shall be bounded in an implementation-defined manner.

2.2.2.227 requested batch service: A service that is either rejected or performed prior to a response from the service to the requester.

2.2.2.228 rerunability: An attribute of a batch job.

If a batch job may be rerun from the beginning after an abnormal termination without affecting the validity of the results, the job is said to be rerunable.

2.2.2.229 restart: Resume the processing of a job from the point of the last checkpoint.

Typically, this is done if the job has been interrupted because of a system failure.

2.2.2.230 service request: A solicitation of services from a client to a server.

A service request may entail the exchange of any number of messages between the client and the server.

In this amendment, the term *request* denotes a service request. When naming specific types of requests, the term *request* is qualified by the type of request, as in Queue Job Request and Delete Job Request.

2.2.2.231 target user: The name of a user on the destination batch server.

The *target user* is the *user_name* under whose account the job is to execute on the destination batch server.

2.2.2.232 submission: The process by which a client requests that a batch server create a job via a Queue Job Request to perform a specified computational task.

⇒ **2.2.3 Abbreviations.** Add the following abbreviations in alphabetic order (disregarding the subclause numbers shown here):

2.2.3.10 BE: *Batch environment*, as defined in Section 8.

2.2.3.11 PBS: An abbreviation for *POSIX batch system* that is used in environment variable names to associate those variables with the batch environment.

⇒ **2.13.2 Symbolic Constants for Portability Specifications.** Add the following constants in alphabetic order to Table 2-19. Add the informative note after the table.

Name	Description
{POSIX2_PBS}	The system supports the Batch Environment Option (see Section 8).
{POSIX2_PBS_ACCOUNTING}	The batch server supports the Job Accounting Option (see 8.2.2.1.1).
{POSIX2_PBS_CHECKPOINT}	The batch server supports the Checkpoint/Restart Option (see 8.2.2.1.2).
{POSIX2_PBS_LOCATE}	The batch server supports the Locate Job Request Option (see 8.2.3.5).
{POSIX2_PBS_MESSAGE}	The batch server supports the Job Message Request Option (see 8.2.3.3).
{POSIX2_PBS_TRACK}	The batch server supports the Track Job Request Option (see 8.2.3.16).

NOTE: The Batch Environment Option is a combination of mandatory and optional services and utilities. The {POSIX_PBS} symbolic constant implies the system supports all the mandatory services and utilities.

Section 8: Batch Environment Option Services and Utilities

This section describes the services and utilities that shall be implemented on all systems that claim conformance to the Batch Environment Option.

8.1 General Concepts

8.1.1 Client-Server Interaction

Batch jobs are created and managed by batch servers. A batch client interacts with a batch server to access batch services on behalf of the user. In order to use batch services, a user must have access to a batch client.

A *batch server* is a computational entity, such as a daemon process, that provides batch services. Batch servers route, queue, modify, and execute batch jobs on behalf of batch clients.

The user utilities described in clauses 8.3 through 8.14 of this amendment are clients of batch services; they allow users to perform actions on the job such as creating, modifying, and deleting batch jobs from a shell command line. Although user utilities may be said to accomplish certain services, they actually obtain services on behalf of a user by means of requests to batch servers.

Client-server interaction takes place by means of the requests defined in this amendment. Because direct access to jobs and queues is limited to servers, clients and servers of different implementations can interoperate, since dependencies on private structures for jobs and queues are limited to servers. Also, batch servers may be clients of other batch servers.

8.1.2 Queues

Two types of batch queues are described in this amendment: routing queues and execution queues.

When a job is placed in a routing queue, it is a candidate for routing.

A job is removed from routing queues under the following conditions:

- The job has been routed to another queue.
- The job has been deleted from the queue.
- The job has been aborted.

When a job is placed in an execution queue, it is a candidate for execution.

A job is removed from an execution queue under the following conditions:

- The job has been executed and exited.
- The job has been aborted.
- The job has been deleted from the queue.
- The job has been moved to another queue.

Access to a queue is limited to the server that manages the queue. Clients never access a queue or a job directly, either to read or write information; all client access to queues or jobs takes place through batch servers.

8.1.3 Job Creation

When a batch server creates a batch job on behalf of a client, it assigns a job identifier to the job. A job identifier consists of both a sequence number that is unique among the sequence numbers issued by that server and the name of the server. Since the server name is unique within a name space, the job identifier is likewise unique within the name space.

The batch server that creates a job returns the server-assigned job identifier to the client that requested the job creation. If the server routes or moves the job to another server, it sends the job identifier with the job. Once assigned, the job identifier of a job never changes.

8.1.4 Job Tracking

Since a job may be moved after creation, the server-name component of the job identifier does not always indicate the location of the job.

An implementation may provide a job tracking mechanism, in which case the user generally does not need to know the location of the job. However, an implementation is not required to provide a job tracking mechanism, in which case the user must find routed jobs by probing the possible destinations.

8.1.5 Job Routing

To route a job, a batch server either moves the job to some other queue that is managed by the server, or requests that some other batch server accept the job.

Each routing queue has one or more queues to which it can route jobs. The batch server administrator creates routing queues.

A batch server may route a job from a routing queue to another routing queue. Servers shall prevent or otherwise handle cases of circular routing paths.

As a deferred service, a batch server routes jobs from the routing queues that it manages. The algorithm by which a batch server selects a queue to which to route a job is implementation defined.

A batch job need not be eligible for routing to all the queues fed by the routing queue from which it is routed. A batch server that has been asked to accept the job may reject the request if the job requires resources that are unavailable to that batch server, or if the client is not authorized to access the server.

Servers may route high-priority jobs before low-priority jobs, but, on other than overloaded systems, the effect may be imperceptible to the user.

If all the servers fed by a routing queue reject requests to accept the job for reasons that are permanent, the batch server that manages the job aborts the job. If all or some rejections are temporary, the server should try to route the job again at some later point.

The conformance document for an implementation shall list the reasons for rejecting the routing of a job. The conformance document shall indicate the reasons for which the routing should be retried later and the reasons for which the job should be aborted.

8.1.6 Job Execution

To execute a job is to create a session leader (a process) that runs the shell program indicated by the *Shell_Path* attribute of the job. The script is passed to the program as its standard input. An implementation of the server may pass the script to the program by other means. The implementation shall document the alternate means in the conformance document. At the time a job begins execution, it is defined to enter the **RUNNING** state. The primary program that is executed by a job is typically, though not necessarily, a shell program.

A batch server executes eligible jobs as a deferred service—no client request is necessary once the job is created and eligible. However, the attributes of a job, such as the job hold type, may render the job ineligible. A batch server scans the execution queues that it manages for jobs that are eligible for execution. The algorithm by which the batch server selects eligible jobs for execution is implementation defined.

As part of creating the process for the job, the batch server opens the standard output and standard error streams of the session.

The attributes of a job may indicate that the server that executes the job is to send mail to a list of users at the time it begins execution of the job.

8.1.7 Job Exit

When the session leader of an executing job terminates, the job exits. As part of exiting a job, the server that manages the job shall remove the job from the queue in which it resides. The server shall transfer output files of the job to a location described by the attributes of the job.

The attributes of a job may indicate that the server that manages the job should send mail to a list of users at the time the job exits.

8.1.8 Job Abort

A batch server aborts jobs for which a required deferred service cannot be performed.

The attributes of a job may indicate that the server that aborts the job shall send mail to a list of users at the time it aborts the job.

8.1.9 Job States

A batch job is always in one of several states: **QUEUED**, **RUNNING**, **HELD**, **WAITING**, **EXITING**, or **TRANSITING**. The state of a job determines the types of requests that the batch server that manages the job can accept for the job. A batch server changes the state of a job either in response to service requests from clients or as a result of deferred services such as job execution or job routing.

A job that is in the **QUEUED** state resides in a queue, but is still pending either execution or routing, depending on the queue type.

A job that resides in an execution queue and is executing is defined to be in the **RUNNING** state. While a job is in the **RUNNING** state, a session leader is associated with the job.

A job that resides in an execution queue, but is ineligible to run because of a hold attribute, is defined to be in the **HELD** state.

A job that is not held, but which must wait until a future date and time before executing, is defined to be in the **WAITING** state.

A job for which the session leader has terminated is defined to be in the **EXITING** state, and the batch server that manages such a job cannot accept job-modification requests that affect the job. While a job is in the **EXITING** state, the batch server that manages the job is staging output files and notifying clients of job completion. Once a job has exited, it no longer exists as an object managed by a batch server.

A job that is being moved from a routing queue to another queue is defined to be in the **TRANSITING** state.

8.1.10 Authorization

In order to access batch services, a user must have execute access to a batch client. For example, to use the command language interface defined in this section, the user must be able to execute the programs that embody those utilities.

Clients, such as the utilities in this section, access batch services by means of requests to one or more batch servers. To acquire the services of any given batch server, the user identifier under which the client runs must be authorized to use that batch server.

The user with an associated user name that creates a job owns the job and can perform actions on the job such as read, modify, delete, and move the job.

A user identifier of the same value at a different host need not be the same user. For example, user name “smith” at host alpha may or may not represent the same person as user name “smith” at host beta. Likewise, the same person may have access to different user names on different hosts.

An implementation may optionally provide an authorization mechanism that permits one user name to access jobs under another user name.

A process on a client host may be authorized to run processes under multiple user names at a server host. Where appropriate, the utilities defined in this amendment provide a means for a user to choose from among such user names when creating or modifying a job.

8.1.11 Batch Administration

The processing of a job by a server is affected by the attributes of the job. The processing of a job may also be affected by the attributes of the queue in which the job resides and by the status of the batch server that manages the job.

A batch administrator is a user that is authorized to modify all the attributes of queues and jobs and to change the status of a server. A batch operator is a user that is authorized to modify some, but not all, of the attributes of jobs and queues, and may change the status of the server.

8.1.12 Notification

Whereas batch servers are persistent entities, clients are often transient. For example, the **qsub** utility creates a batch job and exits. For this reason, batch servers notify users of batch job events by sending mail to the user that owns the job, or to other designated users.

8.2 Batch Services

The presence of Batch Environment Option services is indicated by the configuration variable {POSIX2_PBS}. A conforming batch server provides services as defined in this section.

A batch server provides batch services in two ways:

- The server provides a service at the request of a client.
- The server provides a deferred service as a result of a change in conditions monitored by the server.

If a server cannot complete a request, it rejects the request. If a server cannot complete a deferred service for a job, the server aborts the job.

Table 8-1 is a summary of environment variables that shall be supported by an implementation of the batch server and utilities.

Table 8-1 – Environment Variable Summary

Variable	Description
PBS_DPREFIX	Defines the directive prefix (see 8.14.3)
PBS_ENVIRONMENT	Job is batch or interactive (see 8.2.2.1)
PBS_JOBID	<i>Job_identifier</i> attribute of job (see 8.2.3.8)
PBS_JOBNAME	<i>Job_name</i> attribute of job (see 8.2.3.8)
PBS_O_HOME	Defines the HOME of the client (see 8.14.2)
PBS_O_HOST	Defines the host name of the client (see 8.14.2)
PBS_O_LANG	Defines the LANG of the client (see 8.14.2)
PBS_O_LOGNAME	Defines the LOGNAME of the client (see 8.14.2)
PBS_O_MAIL	Defines the MAIL of the client (see 8.14.2)
PBS_O_PATH	Defines the PATH of the client (see 8.14.2)
PBS_O_QUEUE	Defines the submit queue of the client (see 8.14.2)
PBS_O_SHELL	Defines the SHELL of the client (see 8.14.2)
PBS_O_TZ	Defines the TZ of the client (see 8.14.2)
PBS_O_WORKDIR	Defines the working directory of the client (see 8.14.2)
PBS_QUEUE	Defines the initial execution queue (see 8.2.2.1)

8.2.1 Job States

A batch job is always in one of several states: **QUEUED**, **RUNNING**, **HELD**, **WAITING**, **EXITING**, or **TRANSITING**. The state of a job determines the types of requests that the batch server that manages the job can accept for the job. A batch server changes the state of a job either in response to service requests from clients or as a result of deferred services such as job execution or job routing.

A job that is in the **QUEUED** state resides in a queue but is still pending either execution or routing, depending on the queue type.

A batch server that queues a job in a routing queue shall put the job in the **QUEUED** state. A batch server that puts a job in an execution queue, but has not yet executed the job, shall put the job in the **QUEUED** state.

A job that resides in an execution queue and is executing is defined to be in the **RUNNING** state. While a job is in the **RUNNING** state, a session leader is associated with the job.

A job that resides in an execution queue, but is ineligible to run because of a hold attribute, is defined to be in the **HELD** state.

A job that is not held, but must wait until a future date and time before executing, is defined to be in the **WAITING** state.

When the session leader associated with a running job exits, the job shall be placed in the **EXITING** state.

A job for which the session leader has terminated is defined to be in the **EXITING** state, and the batch server that manages such a job cannot accept job-modification requests that affect the job. While a job is in the **EXITING** state, the batch server that manages the job is staging output files and notifying clients of job completion. Once a job has exited, it no longer exists as an object managed by a batch server.

A job that is being moved from a routing queue to another queue is defined to be in the **TRANSITING** state.

When a job in a routing queue has been selected to be moved to a new destination, then the job is in either the **QUEUED** state or the **TRANSITING** state depending on the server implementation.

Jobs with either a *Execution_Time* attribute value set in the future or a *Hold_Types* attribute of value not equal to **NO_HOLD**, or both, may be routed or held in the routing queue. An implementation shall document the treatment of jobs with the *Execution_Time* or *Hold_Types* attributes in a routing queue.

When a job in a routing queue has not been selected to be moved to a new destination and the job has a *Hold_Types* attribute value of other than **NO_HOLD**, then the job should be in the **HELD** state.

NOTE: The effect of a hold upon a job in a routing queue is implementation defined. The implementation should use the state that matches if the job can route with a hold or not.

When a job in a routing queue has not been selected to be moved to a new destination and the job has a

— *Hold_Types* attribute value of **NO_HOLD**, and a

— *Execution_Time* attribute in the past,

then the job will be in the **QUEUED** state.

When a job in a routing queue has not been selected to be moved to a new destination and the job has a

— *Hold_Types* attribute value of **NO_HOLD**, and a

— *Execution_Time* attribute in the future,

then the job may be in the **WAITING** state.

NOTE: The effect of a future execution time upon a job in a routing queue is implementation defined. The implementation should use the state that matches if the job can route with a hold or not.

Table 8-2 describes the next state of a job, given the current state of the job and the type of request. Table 8-3 describes the response of a batch server to a request, given the current state of the job and the type of request.

8.2.2 Deferred Batch Services

This subclause describes the deferred services performed by batch servers: job execution, job routing, job exit, job abort, and the rerunning of jobs after a restart.

8.2.2.1 Job Execution

To execute a job is to create a session leader (a process) that runs the shell program indicated by the *Shell_Path_List* attribute of the job. The script is passed to the program as its standard input. An implementation of the server may pass the script to the program by other means. The implementation shall document the

Table 8-2 – Next State Table

Request Type	Current State						
	X	Q	R	H	W	E	T
Queue Job Request	Q	e	e	e	e	e	e
Modify Job Request	e	Q	R	H	W	e	T
Delete Job Request	e	X	E	X	X	E	X
Job Message Request	e	Q	R	H	W	E	T
Rerun Job Request	e	e	Q	e	e	e	e
Signal Job Request	e	e	R	H	W	e	e
Job Status Request	e	Q	R	H	W	E	T
Queue Status Request	X	Q	R	H	W	E	T
Server Status Request	X	Q	R	H	W	E	T
Select Jobs Request	X	Q	R	H	W	E	T
Move Job Request	e	Q	R	H	W	e	T
Hold Job Request	e	H	R/H	H	H	e	T
Release Job Request	e	Q	R	Q/W/H	W	e	T
Server Shutdown Request	X	Q	Q	H	W	E	T
Locate Job Request	e	Q	R	H	W	E	T

NOTE: Where

X Nonexistent
 Q **QUEUED**
 R **RUNNING**
 H **HELD**
 W **WAITING**
 E **EXITING**
 T **TRANSITING**
 e Error

alternate means in the conformance document. At the time a job begins execution, it is defined to enter the **RUNNING** state.

A batch server that has an execution queue containing jobs is said to own the queue and manage the jobs in that queue. A batch server that has been started, shall execute the jobs in the execution queues owned by the server. The server shall schedule for execution those jobs in the execution queues that are in the **QUEUED** state. The algorithm for scheduling jobs is implementation defined.

A batch server that executes a job shall create, in the environment of the session leader of the job, an environment variable named **PBS_ENVIRONMENT**, the value of which is the string **PBS_BATCH** encoded in the portable character set.

A batch server that executes a job shall create, in the environment of the session leader of the job, an environment variable named **PBS_QUEUE**, the value of which is the name of the execution queue of the job encoded in the portable character set.

To *rerun* a job is to requeue a job that is currently executing and then kill the session leader of the executing job by sending a SIGKILL prior to completion. See 8.2.3.11. A batch server that reruns a job shall append the standard output and standard error files of the job to the corresponding files of the previous execution, if they exist, with appropriate annotation. If either file does not exist, that file shall be created as in normal execution.

Table 8-3 – Results/Output Table

Request Type	Current State						
	X	Q	R	H	W	E	T
Queue Job Request	O	e	e	e	e	e	e
Modify Job Request	e	O	e	O	O	e	e
Delete Job Request	e	O	O	O	O	e	O
Job Message Request	e	e	O	e	e	e	e
Rerun Job Request	e	e	O	e	e	e	e
Signal Job Request	e	e	O	e	e	e	e
Job Status Request	e	O	O	O	O	O	O
Queue Status Request	O	O	O	O	O	O	O
Server Status Request	O	O	O	O	O	O	O
Select Job Request	e	O	O	O	O	O	O
Move Job Request	e	O	O	O	O	e	e
Hold Job Request	e	O	O	O	O	e	e
Release Job Request	e	O	e	O	O	e	e
Server Shutdown Request	O	O	e	O	O	e	e
Locate Job Request	e	O	O	O	O	O	O

NOTE: Where

O OK

e Error message

The execution of a job by a batch server is controlled by job, queue, and server attributes, as defined in this subclause.

8.2.2.1.1 *Account_Name* Attribute

Accounting is an optional feature of batch servers. If a batch server implements accounting, the statements in this subclause apply and the configuration variable {POSIX2_PBS_ACCOUNTING} shall be set to 1.

A batch server that executes a job shall charge the account named in the *Account_Name* attribute of the job for resources consumed by the job.

If the *Account_Name* attribute of the job is absent from the job attribute list or is altered while the job is in execution, the batch server action is implementation defined.

8.2.2.1.2 *Checkpoint* Attribute

Checkpointing is an optional feature of batch servers. If a batch server implements checkpointing, the statements in this subclause apply and the configuration variable {POSIX2_PBS_CHECKPOINT} shall be set to 1.

There are two attributes associated with the checkpointing feature: *Checkpoint* and *Minimum_Cpu_Interval*. *Checkpoint* is a job attribute, while *Minimum_Cpu_Interval* is a queue attribute. An implementation that does not support checkpointing shall support the *Checkpoint* job attribute to the extent that the server shall maintain and pass this attribute to other servers.

The behavior of a batch server that executes a job for which the value of the *Checkpoint* attribute is CHECKPOINT_UNSPECIFIED is implementation defined. The

implementation shall document the behavior of the batch server.

A batch server that executes a job for which the value of the *Checkpoint* attribute is **NO_CHECKPOINT** shall not checkpoint the job.

A batch server that executes a job for which the value of the *Checkpoint* attribute is **CHECKPOINT_AT_SHUTDOWN** shall checkpoint the job only when the batch server accepts a request to shut down during the time when the job is in the **RUNNING** state.

A batch server that executes a job for which the value of the *Checkpoint* attribute is **CHECKPOINT_AT_MIN_CPU_INTERVAL** shall checkpoint the job at the interval specified by the *Minimum_Cpu_Interval* attribute of the queue for which the job has been selected. The *Minimum_Cpu_Interval* attribute shall be specified in units of CPU minutes.

A batch server that executes a job for which the value of the *Checkpoint* attribute is an unsigned integer shall checkpoint the job at an interval that is the value of either the *Checkpoint* attribute, or the *Minimum_Cpu_Interval* attribute of the queue for which the job has been selected, whichever is greater. Both intervals shall be in units of CPU minutes. When the *Minimum_Cpu_Interval* attribute is greater than the *Checkpoint* attribute, the job shall write a warning message to the standard error stream of the job.

8.2.2.1.3 *Error_Path* Attribute

The *Error_Path* attribute of a running job cannot be changed by a Modify Job Request. When the *Join_Path* attribute of the job is set to the value **FALSE** and the *Keep_Files* attribute of the job does not contain the value **KEEP_STD_ERROR**, a batch server that executes a job shall perform one of the following actions:

- Set the standard error stream of the session leader of the job to the path described by the value of the *Error_Path* attribute of the job.
- Buffer the standard error of the session leader of the job until completion of the job and when the job exits return the contents to the destination described by the value of the *Error_Path* attribute of the job. Where the server buffers standard error is implementation defined.

Applications shall not rely on having access to the standard error of a job prior to the completion of the job.

When the *Error_Path* attribute does not specify a hostname, then the server shall retain the standard error of the job on the host of execution.

When the *Error_Path* attribute does specify a hostname and the *Keep_Files* attribute does not contain the value **KEEP_STD_ERROR**, then the final destination of the standard error of the job shall be on the host whose hostname is specified.

If the path indicated by the value of the *Error_Path* attribute of the job is a relative path, the batch server shall expand the path relative to the home directory of the user on the host to which the file is being returned.

When the server buffers the standard error of the job and the file cannot be opened for write upon completion of the job, then the server shall place the standard error in an implementation-defined location and notify the user of the

location via mail. It shall be possible for the user to process this mail using the **mailx** utility as defined by 4.40.

If a server that does not buffer the standard error cannot open the standard error path of the job for write access, then the batch server shall abort the job.

8.2.2.1.4 *Execution_Time* Attribute

A batch server shall not execute a job before the time represented by the value of the *Execution_Time* attribute of the job. The *Execution_Time* attribute is defined in seconds since the Epoch.

8.2.2.1.5 *Hold_Types* Attribute

A batch server shall support the following hold types:

- S** That can be set or released by a user with at least a privilege level of batch administrator (**SYSTEM**).
- O** That can be set or released by a user with at least a privilege level of batch operator (**OPERATOR**).
- u** That can be set or released by the user with at least a privilege level of user, where the user is defined in the *Job_Owner* attribute (**USER**).
- n** That indicates that none of the *Hold_Types* attributes are set (**NO_HOLD**).

An implementation may define other hold types. The conformance document for an implementation shall describe any additional hold types, how they are specified, their internal representation, their behavior, and how they affect the behavior of other utilities.

The value of the *Hold_Types* attribute shall be the union of the valid hold types (**S**, **O**, **u**, and any implementation-defined hold types) or **n**.

A batch server shall not execute a job if the *Hold_Types* attribute of the job has a value other than **NO_HOLD**. If the *Hold_Types* attribute of the job has a value other than **NO_HOLD**, the job shall be in the **HELD** state.

8.2.2.1.6 *Job_Owner* Attribute

The *Job_Owner* attribute consists of a pair of username and hostname values of the form:

username@hostname

A batch server that accepts a Queue Job Request shall set the *Job_Owner* attribute to a string that is the *username@hostname* of the user who submitted the job.

8.2.2.1.7 *Join_Path* Attribute

A batch server that executes a job for which the value of the *Join_Path* attribute is **TRUE** shall ignore the value of the *Error_Path* attribute and merge the standard error of the job with the standard output of the job.

8.2.2.1.8 *Keep_Files* Attribute

A batch server that executes a job for which the value of the *Keep_Files* attribute includes the value `KEEP_STD_OUTPUT` shall retain the standard output of the job on the host where execution occurs. The standard output shall be retained in the home directory of the user under whose user ID the job is executed and the file name shall be the default file name for the standard output as defined under the `-o` option of the `qsub` utility. The *Output_Path* attribute is not modified.

A batch server that executes a job for which the value of the *Keep_Files* attribute includes the value `KEEP_STD_ERROR` shall retain the standard error of the job on the host where execution occurs. The standard error shall be retained in the home directory of the user under whose user ID the job is executed and the file name shall be the default file name for standard error as defined under the `-e` option of the `qsub` utility. The *Error_Path* attribute is not modified.

A batch server that executes a job for which the value of the *Keep_Files* attribute includes values other than `KEEP_STD_OUTPUT` and `KEEP_STD_ERROR` shall retain these other files on the host where execution occurs. These files shall be retained in the home directory of the user under whose user identifier the job is executed and the file names shall be the default file names for the files as defined in the conformance document for the implementation.

8.2.2.1.9 *Mail_Points* and *Mail_Users* Attributes

A batch server that executes a job for which one of the values of the *Mail_Points* attribute is the value `MAIL_AT_BEGINNING` shall send a mail message to each user account listed in the *Mail_Users* attribute of the job.

The mail message shall contain at least the job identifier, queue, and server at which the job currently resides, and the *Job_Owner* attribute.

8.2.2.1.10 *Output_Path* Attribute

The *Output_Path* attribute of a running job cannot be changed by a Modify Job Request. When the *Keep_Files* attribute of the job does not contain the value `KEEP_STD_OUTPUT`, a batch server that executes a job shall either

- Set the standard output stream of the session leader of the job to the destination described by the value of the *Output_Path* attribute of the job, or
- Buffer the standard output of the session leader of the job until completion of the job and when the job exits return the contents to the destination described by the value of the *Output_Path* attribute of the job.

When the *Output_Path* attribute does not specify a hostname, then the server shall retain the standard output of the job on the host of execution.

When the *Keep_Files* attribute does not contain the value `KEEP_STD_OUTPUT` and the *Output_Path* attribute does specify a hostname, then the final destination of the standard output of the job shall be on the host specified.

If the path specified in the *Output_Path* attribute of the job is a relative path, the batch server shall expand the path relative to the home directory of the user on the host to which the file is being returned.

Whether or not the batch server buffers the standard output of the job until completion of the job is implementation defined. Applications shall not rely on having access to the standard output of a job prior to the completion of the job.

When the server does buffer the standard output of the job and the file cannot be opened for write upon completion of the job, then the server shall place the standard output in an implementation-defined location and notify the user of the location via mail. It shall be possible for the user to process this mail using the `mailx` utility as defined by 4.40.

If a server that does not buffer the standard output cannot open the standard output path of the job for write access, then the batch server shall abort the job.

8.2.2.1.11 *Priority* Attribute

A batch server implementation might choose to preferentially execute a job based on the *Priority* attribute. The interpretation of the job *Priority* attribute by a server is implementation defined. If an implementation uses the *Priority* attribute, it shall interpret larger values of the *Priority* attribute to mean the job shall be preferentially selected for execution.

8.2.2.1.12 *Rerunnable* Attribute

A job that began execution but did not complete because the server either (a) shut down, or (b) terminated abnormally, shall be requeued if the *Rerunnable* attribute of the job has the value `TRUE`.

If a job, which was requeued after beginning execution but prior to completion, has a valid checkpoint file and the server supports checkpointing, then the job shall be restarted from the last valid checkpoint.

If the job cannot be restarted from a checkpoint, then when a job has a rerunnable attribute value of `TRUE` and was requeued after beginning execution but prior to completion, the server shall place the job into execution at the beginning of the job.

When a job has a rerunnable attribute value other than `TRUE` and was requeued after beginning execution but prior to completion, and the job cannot be restarted from a checkpoint, then the server shall abort the job.

8.2.2.1.13 *Resource_List* Attribute

A batch server that executes a job shall establish the resource limits of the session leader of the job according to the values of the *Resource_List* attribute of the job. Resource limits shall be enforced by an implementation-defined method.

8.2.2.1.14 *Shell_Path_List* Attribute

The *Shell_Path_List* job attribute consists of a list of pairs of pathname and host-name values. The host-name component can be omitted, in which case the pathname serves as the default pathname when a server cannot find the name of the host on which it is running in the list.

A batch server that executes a job shall select, from the value of the *Shell_Path_List* attribute of the job, a pathname where the shell to execute the job will be

found. The server shall select the pathname, in order of preference, according to the following methods:

- Select the pathname that contains the name of the host on which the server is running.
- Select the pathname for which the host name has been omitted.
- Select the pathname for the login shell of the user under which the job is to execute.

If the shell path value selected is an invalid pathname, the batch server shall abort the job.

If the value of the selected pathname from the *Shell_Path_List* attribute of the job represents a partial path, the server shall expand the path relative to a path that is implementation defined.

The server that executes the job will execute the program that was selected from the *Shell_Path_List* attribute of the job. The server shall pass the path to the script of the job as the first argument to the shell program.

8.2.2.1.15 *User_List* Attribute

The *User_List* job attribute consists of a list of pairs of user-name and host-name values. The host-name component can be omitted, in which case the user name serves as a default when a server cannot find the name of the host on which it is running in the list.

A batch server that executes a job shall select, from the value of the *User_List* attribute of the job, a user name under which to create the session leader. The server shall select the user name, in order of preference, according to the following methods:

- Select the user name of a value that contains the name of the host on which the server executes.
- Select the user name of a value for which the host name has been omitted.
- Select the user name from the *Job_Owner* attribute of the job.

8.2.2.1.16 *Variable_List* Attribute

A batch server that executes a job shall create, in the environment of the session leader of the job, each environment variable listed in the *Variable_List* attribute of the job, and set the value of each such environment variable to that of the corresponding variable in the variable list.

8.2.2.2 Job Routing

To route a job is to select a queue from a list and move the job to that queue.

A batch server that has routing queues, that have been started, shall route the jobs in the routing queues owned by the server. A batch server is allowed to delay the routing of a job. The algorithm for selecting a job and the queue to which it will be routed is implementation defined.

When a routing queue has multiple possible destinations specified, then the precedence of the destination is implementation defined.

A batch server that routes a job to a queue at another server shall move the job into the target queue with a Queue Job Request.

If the target server rejects the Queue Job Request, the routing server shall retry routing the job or abort the job. A batch server that retries failed routings shall provide a means for the batch administrator to specify the number of retries and the minimum period of time between retries. The means by which an administrator specifies the number of retries and the delay between retries is implementation defined. When the number of retries specified by the batch administrator has been exhausted, the batch server shall abort the job and perform the functions of Job Exit; see 8.2.2.3.

8.2.2.3 Job Exit

For each job in the **EXITING** state, the batch server that exited the job shall perform the following deferred services in the order specified:

- If buffering standard error, move that file into the location specified by the *Error_Path* attribute of the job.
- If buffering standard output, move that file into the location specified by the *Output_Path* attribute of the job.
- If the *Mail_Points* attribute of the job includes **MAIL_AT_EXIT**, send mail to the users listed in the *Mail_Users* attribute of the job. The mail message shall contain at least the job identifier, queue, and server at which the job currently resides, and the *Job_Owner* attribute.
- Remove the job from the queue.

If a server that buffers the standard error output cannot return the standard error file to the standard error path at the time the job exits, the batch server shall either

- Mail the standard error file to the job owner.
- Save the standard error file and mail the location and name of the file where the standard error is stored to the job owner.
- Save the standard error file and notify the user by other means, in which case the conformance document for the implementation shall document the method of notification.

If a server that buffers the standard output cannot return the standard output file to the standard output path at the time the job exits, the batch server shall either

- Mail the standard output file to the job owner.
- Save the standard output file and mail the location and name of the file where the standard output is stored to the job owner.
- Save the standard output file and notify the user by other means, in which case the conformance document for the implementation shall document the method of notification.

591 At the conclusion of job exit processing, the job is no longer managed by a batch
592 server.

593 **8.2.2.4 Batch Server Restart**

594 A batch server that has been either (1) shutdown, or (2) terminated abnormally,
595 and has returned to operation, is said to have restarted.

596 Upon restarting, a batch server shall requeue those jobs managed by the server
597 that were in the **RUNNING** state at the time the server shut down and for which the
598 *Rerunable* attribute of the job has the value **TRUE**.

599 Queues are defined to be nonvolatile. A batch server shall store the content of
600 queues that it controls in such a way that server and system shutdowns do not
601 erase the content of the queues.

602 **8.2.2.5 Job Abort**

603 A batch server that cannot perform a deferred service for a job shall abort the job.

604 A batch server that aborts a job shall perform the following services:

- 605 — Delete the job from the queue in which it resides.
- 606 — If the *Mail_Points* attribute of the job includes the value **MAIL_AT_ABORT**,
607 send mail to the users listed in the value of the *Mail_Users* attribute of the
608 job. The mail message shall contain at least the job identifier, queue, and
609 server at which the job currently resides, the *Job_Owner* attribute, and the
610 reason for the abort.
- 611 — If the job was in the **RUNNING** state, terminate the session leader of the exe-
612 cuting job by sending the session leader a **SIGKILL**, place the job in the
613 **EXITING** state, and perform the services of Job Exit.

614 **8.2.3 Requested Batch Services**

615 This subclause describes the services provided by batch servers in response to
616 requests from clients. Table 8-4 summarizes the current set of batch service
617 requests and for each gives its type (deferred or not) and whether it is an optional
618 function.

619 If a request is rejected because the client is not authorized to perform the action,
620 the batch server shall return the same status as when the job does not exist.

621 **8.2.3.1 Delete Job Request**

622 A job is defined to have been deleted when it has been removed from the queue in
623 which it resides and not instantiated in another queue. A client requests that the
624 server that manages a job delete the job. Such a request is called a Delete Job
625 Request.

Table 8-4 – Batch Services Summary

Batch Service	Deferred	Optional
Job Execution	yes	no
Job Routing	yes	no
Job Exit	yes	no
Batch Server Restart	yes	no
Job Abort	yes	no
Queue Job Request	no	no
Modify Job Request	no	no
Delete Job Request	no	no
Job Message Request	no	yes
Rerun Job Request	no	no
Signal Job Request	no	no
Job Status Request	no	no
Queue Status Request	no	no
Server Status Request	no	no
Select Jobs Request	no	no
Move Job Request	no	no
Hold Job Request	no	no
Release Job Request	no	no
Server Shutdown Request	no	no
Locate Job Request	no	yes
Track Job Request	no	yes

A batch server shall reject a Delete Job Request if any of the following statements are true:

- The user of the client is not authorized to delete the designated job.
- The designated job is not managed by the server.
- The designated job is in a state inconsistent with the delete request.

A batch server may reject a Delete Job Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Delete Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server requested to delete a job shall delete the job if the job exists and is not in the **EXITING** state.

A batch server that deletes a job in the **RUNNING** state shall send a SIGKILL signal to the session leader of the job. A batch server may send additional signals to the session leader of job prior to sending the SIGKILL signal. The conformance document for such a batch server shall document the signals that are sent to the session leader.

A batch server that deletes a job in the **RUNNING** state shall place the job in the **EXITING** state after it has killed the session leader of the job and shall perform the services of Job Exit.

8.2.3.2 Hold Job Request

A client can request that the server add one or more holds to a job. Such a request is called a Hold Job Request.

A batch server shall reject a Hold Job Request if any of the following statements are true:

- The batch server does not support one or more of the requested holds to be added to the job.
- The user of the client is not authorized to add one or more of the requested holds to the job.
- The batch server does not manage the specified job.
- The designated job is in the **EXITING** state.

A batch server may reject a Hold Job Request for other reasons. The conformance document for an implementation shall document the reasons for which a Hold Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A server that accepts a Hold Job Request for a job in the **RUNNING** state shall place a hold on the job. The conformance document shall describe what effect, if any, the hold will have on a job in the **RUNNING** state.

A batch server that accepts a Hold Job Request shall add each type of hold listed in the Hold Job Request, that is not already present, to the value of the *Hold_Types* attribute of the job.

8.2.3.3 Job Message Request

Job Message Request is an optional feature of batch servers. If an implementation supports Job Message Request, the statements in this subclause apply and the configuration variable {POSIX2_PBS_MESSAGE} shall be set to 1.

A client can request that a batch server write a message into certain output files of a job. Such a request is called a Job Message Request.

A batch server shall reject a Job Message Request if any of the following statements are true:

- The batch server does not support sending messages to jobs.
- The user of the client is not authorized to post a message to the designated job.
- The designated job does not exist on the server.
- The designated job is not in the **RUNNING** state.

A batch server may reject a Job Message Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Job Message Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a Job Message Request shall write the message sent by the client into the files indicated by the client.

8.2.3.4 Job Status Request

A client can request that a batch server respond with the status and attributes of a job. Such a request is called a Job Status Request.

A batch server shall reject a Job Status Request if any of the following statements are true:

- The user of the client is not authorized to query the status of the designated job.

- The designated job is not managed by the server.

A batch server may reject a Job Status Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Job Status Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a request for job status shall return a Job Status Message to the client.

A batch server may return other information in response to a Job Status Request.

8.2.3.5 Locate Job Request

Locate Job Request is an optional feature of batch servers. If an implementation supports Locate Job Request, the statements in this subclause apply and the configuration variable {POSIX2_PBS_LOCATE} shall be set to 1.

A client can ask a server to respond with the location of a job that was created by the server. Such a request is called a Locate Job Request.

A server that accepts a Locate Job Request shall return a Job Location message to the client.

A batch server may reject a Locate Job Request for a job that was not created by that server.

A batch server may reject a Locate Job Request for a job that is no longer managed by that server—that is, for a job that is not in a queue owned by that server.

A batch server may reject a Locate Job Request for other reasons. The conformance document for an implementation shall document the reasons for which a Locate Job Request may be rejected.

8.2.3.6 Modify Job Request

Batch clients modify (alter) the attributes of a job by making a request to the server that manages the job. Such a request is called a Modify Job Request.

A batch server shall reject a Modify Job Request if any of the following statements are true:

- The user of the client is not authorized to make the requested modification to the job.
- The designated job is not managed by the server.
- The requested modification is inconsistent with the state of the job.
- An unrecognized resource is requested for a job in an execution queue.

A batch server may reject a Modify Job Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Modify Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a Modify Job Request shall modify all the specified attributes of the job. A batch server that rejects a Modify Job Request shall modify none of the attributes of the job.

If the servicing by a batch server of an otherwise valid request would result in no change, then the server shall indicate successful completion of the request.

8.2.3.7 Move Job Request

A client can request that a batch server move a job to another destination. Such a request is called a Move Job Request.

A batch server shall reject a Move Job Request if any of the following statements are true:

- The user of the client is not authorized to remove the designated job from the queue in which the job resides.
- The user of the client is not authorized to move the designated job to the destination.
- The designated job is not managed by the server.
- The designated job is in the **EXITING** state.
- The destination is inaccessible.

A batch server can reject a Move Job Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Move Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a Move Job Request shall perform the following services:

- Queue the designated job at the destination.
- Remove the designated job from the queue in which the job resides.

If the destination resides on another server, the server shall queue the job at the destination by sending a Queue Job Request to the other server. If the Queue Job Request fails, the server shall reject the Move Job Request. If the Queue Job Request succeeds, the server shall remove the job from its queue.

The server shall not modify any attributes of the job.

8.2.3.8 Queue Job Request

A batch queue is controlled by one and only one batch server. A batch server is said to own the queues that it controls. Batch clients make requests of batch servers to have jobs queued. Such a request is called a Queue Job Request.

A batch server requested to queue a job for which the queue is unspecified shall select a queue for the job. Such a queue is called the default queue of the server. The conformance document for the implementation shall document the means by which the server determines the default queue. The implementation shall provide the means for a batch administrator to specify the default queue. The queue, whether specified or defaulted, is called the target queue.

A batch server shall reject a Queue Job Request if any of the following statements are true:

- The client is not authorized to create a job in the target queue.
- The request specifies a queue that does not exist on the server.
- The target queue is an execution queue and the server cannot satisfy a resource requirement of the job.
- The target queue is an execution queue and an unrecognized resource is requested.
- The target queue is an execution queue, the batch server does not support checkpointing, and the value of the *Checkpoint* attribute of the job is not **NO_CHECKPOINT**.
- The job requires access to a user identifier that the client is not authorized to access.

A batch server may reject a Queue Job Request for other reasons. The conformance document for an implementation shall document the reasons for which a Queue Job Request may be rejected.

A batch server that accepts a Queue Job Request for a job for which the **PBS_O_QUEUE** value is missing from the value of the *Variable_List* attribute of the job shall add that variable to the list and set the value to the name of the target queue. Once set, no server shall change the value of **PBS_O_QUEUE**, even if the job is moved to another queue.

A batch server that accepts a Queue Job Request for a job for which the **PBS_JOBID** value is missing from the value of the *Variable_List* attribute shall add that variable to the list and set the value to the *job_identifier* assigned by the server in the format: *sequence_number.server*.

A batch server that accepts a Queue Job Request for a job for which the **PBS_JOBNAME** value is missing from the value of the *Variable_List* attribute of

826 the job shall add that variable to the list and set the value to the *Job_Name* attri-
827 bute of the job.

828 **8.2.3.9 Queue Status Request**

829 A client can request that a batch server respond with the status and attributes of
830 a queue. Such a request is called a Queue Status Request.

831 A batch server shall reject a Queue Status Request if any of the following state-
832 ments are true:

833 — The user of the client is not authorized to query the status of the desig-
834 nated queue.

835 — The designated queue does not exist on the server.

836 A batch server may reject a Queue Status Request for other reasons. The confor-
837 mance document for an implementation shall describe the reasons for which a
838 Queue Status Request is rejected. The conformance document for an implementa-
839 tion shall describe the method used to determine whether or not the user of a
840 client is authorized to perform the requested action.

841 A batch server that accepts a Queue Status Request shall return a Queue Status
842 Reply to the client.

843 **8.2.3.10 Release Job Request**

844 A client can request that server remove one or more holds from a job. Such a
845 request is called a Release Job Request.

846 A batch server shall reject a Release Job Request if any of the following state-
847 ments are true:

848 — The user of the client is not authorized to remove one or more of the
849 requested holds from the job.

850 — The batch server does not manage the specified job.

851 A batch server may reject a Release Job Request for other reasons. The confor-
852 mance document for an implementation shall document the reasons for which a
853 Release Job Request may be rejected. The conformance document for an imple-
854 mentation shall describe the method used to determine whether or not the user of
855 a client is authorized to perform the requested action.

856 A batch server that accepts a Release Job Request shall remove each type of hold
857 listed in the Release Job Request, that is present, from the value of the *Hold_-*
858 *Types* attribute of the job.

859 **8.2.3.11 Rerun Job Request**

860 To rerun a job is to kill the session leader of the job and leave the job eligible for
861 re-execution. A batch client can request that a server rerun a job. Such a request
862 is called Rerun Job Request.

A batch server shall reject a Rerun Job Request if any of the following statements are true:

- The user of the client is not authorized to rerun the designated job.
- The *Rerunable* attribute of the designated job has the value **FALSE**.
- The designated job is not in the **RUNNING** state.
- The server does not manage the designated job.

A batch server may reject a Rerun Job Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Rerun Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that rejects a Rerun Job Request shall in no way modify the execution of the job.

A batch server that accepts a request to rerun a job shall perform the following services:

- Requeue the job in the execution queue in which it was executing.
- Send a SIGKILL signal to the process group of the session leader of the job.

An implementation may indicate to the job owner that the job has been rerun. The conformance document for an implementation shall state whether the job owner is notified that a job is rerun, and if so, shall describe the means used.

A batch server that reruns a job may send other signals to the session leader of the job prior to sending the SIGKILL signal. The conformance document for an implementation shall describe any other signals that may be sent.

A batch server may preferentially select a rerun job for execution. The conformance document for an implementation shall state whether or not rerun jobs will be selected for execution before other jobs.

8.2.3.12 Select Jobs Request

A batch client can request from a server a list of jobs managed by that server that match a list of selection criteria. Such a request is called a Select Jobs Request. All the jobs managed by the server that receives the request are candidates for selection.

A batch server that accepts a Select Jobs Request shall return a list of zero or more job identifiers that correspond to jobs that meet the selection criteria.

If the client is not authorized to query the status of a job, the server shall not select the job.

8.2.3.13 Server Shutdown Request

A server is defined to have shut down when it does not respond to requests from clients and does not perform deferred services for jobs. A client can request that a batch server shut down. Such a request is called a Server Shutdown Request.

A batch server shall reject a Server Shutdown Request from a client that is not authorized to shut down the server. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server may reject a Server Shutdown Request for other reasons. The conformance document for an implementation shall document the reasons for which a Server Shutdown Request may be rejected.

At server shutdown, a server shall do, in order of preference, one of the following:

- If checkpointing is implemented and the job is checkpointable, then checkpoint the job and requeue it.
- If the job is rerunnable, then requeue the job to be rerun (restarted from the beginning).
- Abort the job.

8.2.3.14 Server Status Request

A client can request that a batch server respond with the status and attributes of the server. Such a request is called a Server Status Request.

A batch server shall reject a Server Status Request if the following statement is true:

- The user of the client is not authorized to query the status of the designated server.

A batch server may reject a Server Status Request for other reasons. The conformance document for an implementation shall describe the reasons for which a Server Status Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a Server Status Request shall return a Server Status Reply to the client.

8.2.3.15 Signal Job Request

A batch client can request that a batch server signal the session leader of a job. Such a request is called a Signal Job Request.

A batch server shall reject a Signal Job Request if any of the following statements are true:

- The user of the client is not authorized to signal the job.
- The job is not in the **RUNNING** state.
- The server does not manage the designated job.
- The requested signal is not supported by the operating system.

A batch server may reject a Signal Job Request for other reasons. The conformance document for an implementation shall describe the reasons for which a

Signal Job Request may be rejected. The conformance document for an implementation shall describe the method used to determine whether or not the user of a client is authorized to perform the requested action.

A batch server that accepts a request to signal a job shall send the signal requested by the client to the process group of the session leader of the job.

8.2.3.16 Track Job Request

Track Job Request is an optional feature of batch servers. If an implementation supports Track Job Request, the statements in this subclause apply and the configuration variable {POSIX2_PBS_TRACK} shall be set to 1.

Track Job Request provides a method for tracking the current location of a job. Clients may use the tracking information to determine the server that should receive a batch server request.

If Track Job Request is supported by a server, then when the server queues a job as a result of a Queue Job Request, and the server is not the server that created the job, the server shall send a Track Job Request to the server that created the job.

If Track Job Request is supported by a server, then the Track Job Request may also be sent to other servers as a backup to the primary server. The method by which backup servers are specified is implementation defined.

If Track Job Request is supported by a server that receives a Track Job Request, then the server shall record the current location of the job as contained in the request.

8.3 Common Behavior for Batch Environment Option Utilities

8.3.1 Job Identifier

A utility shall recognize *job_identifiers* of the format:

sequence_number[@*server_name*]

where:

sequence_number An integer that when combined with *server_name* provides a job identifier that is unique within the batch system.

server_name The name of the batch server to which the job was originally submitted.

server The name of the batch server that is currently managing the batch job.

If the application omits the *server_name* portion of a job identifier, a utility shall use the name of a default server.