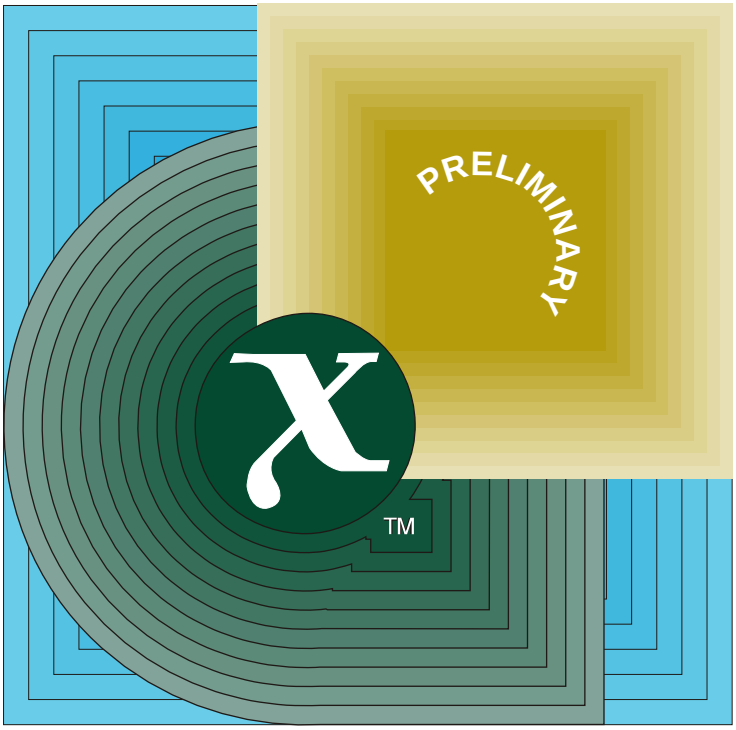
Preliminary Specification

Distributed Audit Service (XDAS)



THE Open GROUP

[This page intentionally left blank]

Preliminary Specification

Distributed Audit Service (XDAS)

The Open Group

Preliminary Specification

Distributed Audit Service (XDAS)

ISBN: 1-85912-139-X
Document Number: P441

Published in the U.K. by The Open Group, January 1998.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Contents

List of Figures

List of Tables

Preface

The Open Group

The Open Group, a vendor and technology-neutral consortium, is committed to delivering greater business efficiency by bringing together buyers and suppliers of information technology to lower the time, cost, and risks associated with integrating new technology across the enterprise.

The Open Group's mission is to offer all organizations concerned with open information infrastructures a forum to share knowledge, integrate open initiatives, and certify approved products and processes in a manner in which they continue to trust our impartiality.

In the global eCommerce world of today, no single economic entity can achieve independence while still ensuring interoperability. The assurance that products will interoperate with each other across differing systems and platforms is essential to the success of eCommerce and business workflow. The Open Group, with its proven testing and certification program, is the international guarantor of interoperability in the new century.

The Open Group provides opportunities to exchange information and shape the future of IT. The Open Group's members include some of the largest and most influential organizations in the world. The flexible structure of The Open Group's membership allows for almost any organization, no matter what their size, to join and have a voice in shaping the future of the IT world.

More information is available on The Open Group web site at *http://www.opengroup.org*.

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification. The Open Group portfolio of test suites includes the *Westwood* family of tests and the associated certification program for Version 3 of the Single UNIX Specification, as well tests for CDE, CORBA, Motif, Linux, LDAP, POSIX.1, POSIX.2, POSIX Realtime, Sockets, UNIX, XPG4, XNFS, XTI, and X11. The Open Group test tools are essential for proper development and maintenance of standards-based products, ensuring conformance of products to industry-standard APIs, applications portability, and interoperability. In-depth testing identifies defects at the earliest possible point in the development cycle, saving costs in development and quality assurance.

More information is available at *http://www.opengroup.org/testing*.

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available at *http://www.opengroup.org/pubs*.

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards-compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such,

both previous and new documents are maintained as current publications.

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published at *http://www.opengroup.org/corrigenda*.

Full catalog and on-line ordering information on all Open Group publications is available at *http://www.opengroup.org/pubs*.

This Document

This document is a Preliminary Specification (see above).

- Chapter 1 is an introduction to the XDAS.
- Chapter 2 is a conformance statement.
- Chapter 3 describes the audit service model.
- Chapter 4 defines the logical data structures used within this specification.
- Chapter 5 provides an overview of the functions defined by this specification and how they are used.
- Chapter 6 describes the parameters required by the XDAS API,
- Chapter 7 describes the XDAS API function definitions,
- Appendix A provides a mapping of domain specific events to the generic set of event classes identified within this specification,
- A glossary of terms used within this specification is provided.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for filenames, and C-language keywords, type names, data structures and their members.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote:
  - — C-language variable names, for example, substitutable argument prototypes
  - — C-language functions; these are shown as follows: *name*( ).
- Normal font is used for the names of constants and literals.
- The notation **<file.h>** indicates a header file.
- The notation [EABCD] is used to identify a C-language return code EABCD.
- Syntax, code examples and user input in interactive examples are shown in `fixed width` font.
- Variables within syntax statements are shown in *`italic fixed width font`*.
- Language-independent functions and arguments use ***bold italic*** font, for example, ***function***( ) and ***argument***.

Trade Marks

The Open Group and Boundaryless Information Flow are trademarks and UNIX is a registered trademark of The Open Group in the United States and other countries. All other trademarks are the property of their respective owners.

Oracle® is a registered trademark of Oracle Corporation.

Solaris® is a registered trademark of Sun Microsystems, Inc.

Acknowledgements

The OpenGroup gratefully acknowledges the work of the OpenGroup Security Program Group in the development of this specification.

Referenced Documents

The following documents are referenced in this specification:

Internet RFC 2044
: Internet RFC 2044 UTF-8, a transformation format of Unicode and ISO 10646.

CESG Memo
: CESG Memorandum No.1 Issue 1.2 Oct 1992, Glossary of Security Terminology.

Federal Criteria
: Federal Criteria Version 1.0 Dec 1992, Federal Criteria for Information Technology Security.

ISO / IEC 7498-2
: ISO / IEC 7498-2: 1989, Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.

ISO / IEC 10181
: ISO / IEC 10181, Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems —

  10181-1: Part 1: Security Frameworks Overview
  10181-2: Part 2: Authentication Framework
  10181-3: Part 3: Access Control
  10181-4: Part 4: Non-repudiation Framework
  10181-5: Part 5: Integrity Framework
  10181-6: Part 6: Confidentiality Framework
  10181-7: Part 7: Security Audit Framework

ITSEC
: Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria, June 1991, Version 1.2, published by the Commission of the European Communities.

POSIX.0
: IEEE Std 1003.0 / D15, June 1992, Draft Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 0.

X.509
: ISO / IEC 9594-8: 1990, Information Technology — Open Systems Interconnection — The Directory — Part 8: Authentication Framework, together with:

  Technical Corrigendum 1: 1991 to ISO / IEC 9594-8: 1990.

ISO 8859-1:1987 Information processing -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1
:

The following Open Group documents are referenced in this specification:

XDSF
: Guide, December 1994, Distributed Security Framework (ISBN: 1-85912-071-7, G410), published by The Open Group.

Federated Naming
: Preliminary Specification, August 1994, Federated Naming: The XFN Specification, (ISBN: 1-85912-458-8, P403), published by The Open Group.

XEMS

Preliminary Specification, June 1996, Systems Management: Event Management Service, Draft V0.3

Chapter 1

Introduction

The purpose of security audit services is to provide support for:

- the principle of accountability, that is holding users of a system accountable for their actions within the system, and
- detection of security policy violations, that is the detection of attempts by unauthorized individuals to access the system and of attempts by authorized users to misuse their access to the system.

Many components of distributed systems now include some form of security auditing or event logging capability whereby the component records events deemed to have security relevance within the domain of that component. These services are provided via component specific interfaces and use component specific audit record formats.

However, within distributed systems security relevant activity is not isolated within individual components but spans many components. For example, an intrusion attempt may be made via multiple entry points to the distributed system. Such attempts are not necessarily focused through single points of entry. Also the purpose of a distributed system is to enable the end-users of the system to utilize the resources of components throughout the system and not just those of their local workstation.

Within a distributed system it is therefore necessary to monitor activity across and between components. This is made difficult by the current component specific approaches. It is not easy to compare activity across system components when the events monitored and the record formats may be different. It is especially difficult to do this in a timely manner to detect and respond to intrusion attempts.

The objective of the XDAS specification is to define:

- a set of generic events of relevance at a global distributed system level. For example, end-user system sign-on and the initiation and termination of communication sessions.
- a common portable audit record format to facilitate the merging and analysis of audit information from multiple components at the distributed system level.
- an API for use by applications to submit events to XDAS.
- an API to import audit data from existing component specific audit services to XDAS.
- an API to configure event preselection criteria for event submission to XDAS.
- an API to read records from a XDAS audit trail

This service is intended to be a complement to existing system component specific audit services, not to replace them. Such local audit services are also likely to handle events, and a level of detail, that may be irrelevant at the global level of XDAS.

Interfaces are supported for use by four different types of applications:

- an API to submit events to the audit service, for use by applications that generate audit records and use XDAS to log such events.
- an API and a common audit event record format for use by existing component specific audit services to import audit records into the XDAS audit stream for distributed system level analysis.

- an API to support the configuration of event preselection criteria and event disposition actions, for use by XDAS audit event management applications.
- an API together with a common audit event record format, for use by Audit Log Analysis applications.

The XDAS-API provides the following benefits:

- Application developers have a common API, a generic set of audit events, and a common audit format regardless of the platform on which the XDAS service is running. This is of benefit to the developers of both applications that detect and wish to record security relevant events and of applications that analyze audit events.
- Platform and application infrastructure vendors are able to support the needs of users at the distributed system level within a heterogeneous environment without the necessity to re-engineer their current operating system or application specific audit service implementations, perhaps with resulting performance implications.
- End-user organizations benefit through increased effectiveness in enforcing individual accountability within a distributed environment.

1.1 Business Requirements

The following business requirements for a distributed audit service have been identified. They are detailed in full in this section to convey the overall service context that XDAS is intended to be capable of supporting. The current scope of the XDAS specification is not intended to encompass all these requirements but to provide a basic level of service on which the more complete requirements may be eventually satisfied by developing applications to utilize the XDAS functions. The requirements are grouped according to audit event services, audit service management, audit event management, audit log management and audit event enquiry facilities.

1.1.1 Audit Event Services

Security events are detected outside the XDAS by an operating system or applications. The requirements on a distributed audit service are as follows

- Handle event records newly generated at the local API level.
- Support the preselection of criteria for the submission of an event, thereby reducing the numbers of audit events generated and analyzed.
- Filter and analyze records for instances or accumulations of pre-determined security events, and trigger timely notification. These filters shall be driven by parameters in a standard format. Three types of event or compound event are identified:
  - — a single record selected by one or more fields
  - — sequences of selected records
  - — timed sequences of records
- Generate local alarms.
- Generate messages to be passed to the audit service management interface.
- Take pre-defined action on the occurrence of specific events.

- Receive records passed on from another system in a standard format and re-interpret them in the context of extra information available from event records arriving from other systems.

Of these requirements, the scope of this XDAS specification provides support for:

- The submission of audit events via an API.
- The return, via the API, to calling applications of the result of applying preselection criteria.
- The capability for the definition of basic forms of event disposition (e.g., log, alarm, action) which can be acted upon by an audit analysis application to meet the other requirements defined above.

1.1.2 Audit Service Management

The business requirements for the user interface for managing the audit service are:

- Support a consistent management interface.
- Integrate the audit system management interface with other elements in the system management infrastructure, including logs, protocols and databases and the management of authorizations.
- Support both Remote and Local Administration
  The XDAS must support role-based decentralized administration, such that individuals are only presented with the data that apply to their area of responsibility.
- Support both equivalent GUI and command line access so that the functions are available regardless of the mode of interaction.

None of these requirements are within the scope of this XDAS specification. However, it does define an API for audit event filter management and defines an authorization model that can be used to support role-based access control. This facilitates the development of management applications to meet the above requirements.

1.1.3 Audit Event Management

The following are requirements on the Audit Event Management interface:

- Support the configuration of the disposition of audit alarms, such that audit events of a specific source and type can be sent to a particular destination, and to a particular role at that destination to be actioned.
- Provide a set of standard calls to modify the parameters which define the filtering performed. These are used to configure the actions taken by the filtering and analysis component on each system. They may be originated by an operator or automatically as a result of event processing.
- Support two types of configuration: *static configuration* and *dynamic configuration*.

  With *static configuration*, the levels of audit data to be generated are pre-set by operator intervention. With *dynamic configuration*, the events or series of events detected are used to re-configure the filters on the monitor. Reconfiguration can involve increasing or decreasing the level of monitoring activity, as deemed appropriate by the analysis of the event or series of events.
- Determine and effect change to the configuration of security event detection on each of the platforms in a distributed environment. If several systems are monitored and all have a common requirement for maintaining a particular level of event logging, then a single definition should be applied to all.

- Be able to record a security event message whenever a change to the configuration of the event discrimination service is made.

The scope of the XDAS specification does not directly include any of the above functionality. It is expected that much of this functionality may eventually be included within implementations. However, interfaces to event management services are being defined in other specifications, (see XEMS). Chapter 3 includes an example of how an XDAS system may be implemented over an event management service.

This XDAS specification defines an API including functions for the definition, enabling and disabling of filtering criteria and the definition of the disposition of events based on those criteria. This API may be used to develop applications that provide the higher level services described above.

1.1.4 Audit Log Management

Audit Log Management requirements are:

- Log records to a protected audit record repository.
- Ensure that the sequence of events recorded is a reflection of what actually transpired. Thus, any mechanism which generates audit data should incorporate a *header* or common set of data which is co-ordinated with other systems with which it interacts. The header should contain a minimum set of information describing the date, time, location, initiator, target, message, etc., of the activity. Platforms, applications and network services shall have the ability to add domain specific information to the information set.

The scope of this XDAS specification includes the definition of the contents of an audit record including the provision for domain specific information for the purposes of analysis and the exchange and merging of audit event records. The internal format of audit logs and interfaces for the management of those logs are not within the scope of this version of XDAS.

1.1.5 Audit Event Enquiry

The Audit Event Enquiry requirements are:

- Define a common format for audit events for use by analysis applications.

A common format for audit events is defined by this XDAS specification.

1.2 Functional Scope of XDAS Specification

This subsection summarizes the functions that are within scope and out of scope for this version of the XDAS specification.

1.2.1 Within Scope

The XDAS provides a set of primitives only, which are used by audit applications. This version of the XDAS specification provides support for:

- The submission of audit events by an application via an API.
- The import of audit events from an existing audit service via an API.
- The return to calling applications of the result of applying preselection criteria for both the event submission and event import APIs.
- An API for the definition of audit event filters defining preselection criteria and basic forms of event disposition (e.g., log, alarm, action) which can be acted upon by a audit analysis applications.
- An authorization model that can be used to support role-based access control. This facilitates the development of management applications supporting roles and subdivision of duties.
- Defines a common format for audit events for use by analysis applications. This format includes both basic audit information and provision for domain specific information. This will facilitate the development of audit analysis applications including the exchange and merging of audit event records within distributed environments.

1.2.2 Out of Scope

The following facilities and services are deemed to be out of scope.

- Event Detection
  The detection of security relevant events is done outside the audit service. The specification assumes that that the applications responsible for even detection will prevent any unauthorized modification of those event detection services.
- Audit Filter Propagation
  XDAS defines interfaces for the creation and management of audit filters. This version of the specification does not define any protocols or data formats for the propagation of those filters between XDAS components.
- Detection of sequences of events or compound events
  XDAS provides the basic functionality for the submission and filtering of individual events together with a common audit event record format for audit event consolidation and analysis. An application capable of detecting complex sequences of events or combinations of events can be implemented over these basic XDAS services.
- Dynamic Modification of Audit Filter Parameters
  XDAS does not include functionality for the analysis of monitored security related events to determine whether modifications are needed to the filter parameters. This functionality falls within the scope of an audit administration application that can be implemented over the XDAS services provided.
- Domain Specific Event
  XDAS is not attempting to map all operating system or domain specific events to XDAS generic events, only those of significance at a distributed system level.
- Graphical User Interface (GUI)
  The specification provides functions that may support the construction of GUI tools but does not address the definition of these tools.

- Audit Log Analysis
  The XDAS provides a set of interfaces for audit log analysis. It does not support queries on the audit log against a set of selection criteria. Nor does it define any of the audit log analysis tools.

  It is assumed that the audit analysis tools will consolidate recorded security related events as part of their analysis of the audit logs.

- Audit Log Management
  The current XDAS specification views the audit log as a stream of time ordered audit event records. No management structure is imposed on this stream and no functions are specified for the management of the system resources, for example files, used for the storage and processing of the stream.

1.3 Security Requirements

An implementation of the XDAS needs to meet the following security requirements:

- Prevent unauthorized recording of audit event records
- Prevent unauthorized modification of the audit service configuration data.
- Prevent unauthorized modification of the event detection records.
- Prevent unauthorized disclosure of the event records.
- Support adequate separation of duties for users.
- Provide appropriate measures in dealing with an unauthorized denial of service, for example, by suspending an offending process, if appropriate.
- Protect audit service configuration data.
- Protect the *audit log* and its contents from any unauthorized modification or deletions.
- Protect the audit log by making it accessible only to principals acting in specific administrative or security roles.

The security requirements shall be met by using underlying distributed system security services and platform security services, wherever possible.

1.4 Non-functional Requirements

The following non-functional requirements have been taken as input to this specification:

- the XDAS shall be application independent
- the XDAS shall not impose a particular placement of access control to distributed audit services within an operating system kernel
- The XDAS shall not constrain future extensibility. Nor shall it constrain the services of other audit systems, including operating system and site specific events types and associated data.

Chapter 2

Conformance Statement

The following XDAS implementation conformance categories are defined:

- **Basic XDAS Conformance**
  This is applicable to an implementation of XDAS that supports the Common Audit Record Format and the Audit Read API in support of Audit Trail Analysis Applications. All implementations shall comply with this basic conformance criteria.
- **XDAS Import API Option Conformance**
  This is applicable to an implementation of XDAS that supports the Audit Log Import API.
- **XDAS Event Submission API Option Conformance**
  This is applicable to an implementation of XDAS that supports the Audit Event Service Client API for direct use by applications.
- **XDAS Filter Management API Option Conformance**
  This is applicable to an implementation of XDAS that supports a filtering capability and the Audit Event Management API.

Note: Early implementations are expected to be by vendors with existing audit functionality who will probably limit their scope to Basic XDAS Conformance and XDAS Import API Conformance to provide support for the common audit record format.

2.1 Basic XDAS Conformance

An implementation of XDAS that conforms with this conformance category shall support the following interfaces:

```
xdas_close_audit_stream     xdas_get_next
xdas_initialize_session     xdas_open_audit_stream
xdas_parse_record           xdas_release_buffer
xdas_rewind_audit_stream    xdas_terminate_session
```

2.2 XDAS Import API Option Conformance

An implementation of XDAS that conforms with this conformance category shall support the following interfaces in addition to those defined for Basic XDAS Conformance:

```
xdas_import_event_records
```

2.3 XDAS Event Submission API Option Conformance

An implementation of XDAS that conforms with this conformance category shall support the following interfaces in addition to those defined for Basic XDAS Conformance:

```
xdas_commit_record      xdas_discard_record
xdas_put_event_info     xdas_start_record
xdas_timestamp_record
```

2.4 XDAS Filter Management API Option Conformance

An implementation of XDAS that conforms with this conformance category shall support the following interfaces in addition to those defined for Basic XDAS Conformance:

```
xdas_create_filter          xdas_delete_filter
xdas_disable_filter         xdas_enable_filter
xdas_get_filter             xdas_list_filters
xdas_release_filter_list
```

2.5 Requirements on an Implementation

- An implementor is required to describe the basis on which, and the extent to which, the authorization model defined within XDAS is implemented and how the user assigns XDAS authorizations to callers. For example, an early implementation may restrict itself to mapping all XDAS authorizations to UNIX superuser privilege.
- An XDAS implementation shall not hinder the achievement of adequate performance over the network.
- An XDAS implementation shall utilize trustworthy universal timestamps on event records if available. Because the XDAS cannot assume a trusted time service is available, provision is included within the XDAS audit record format to include a measure of the uncertainty of the time at which the recorded event occurred. This uncertainty information needs to be inserted into the records when they are imported to or exchanged between XDAS systems.
- An implementor is required to state if the audit IDs are distinguished from access IDs and principal IDs.
- An implementor is required to document how additional registered audit event numbers are configured into an XDAS service.

- An implementation of the XDAS needs to meet the following security requirements:
  - — Prevent unauthorized recording of audit event records
  - — Prevent unauthorized modification of the audit service configuration data.
  - — Prevent unauthorized modification of the event detection records.
  - — Prevent unauthorized disclosure of the event records.
  - — Support adequate separation of duties for users.
  - — Provide appropriate measures in dealing with an unauthorized denial of service, for example, by suspending an offending process, if appropriate.
  - — Protect audit service configuration data.
  - — Protect the *audit log* and its contents from any unauthorized modification or deletions.
  - — Protect the audit log by making it accessible only to principals acting in specific administrative or security roles.

  The security requirements shall be met by using underlying distributed system security services and platform security services, wherever possible.

3.1 Introduction

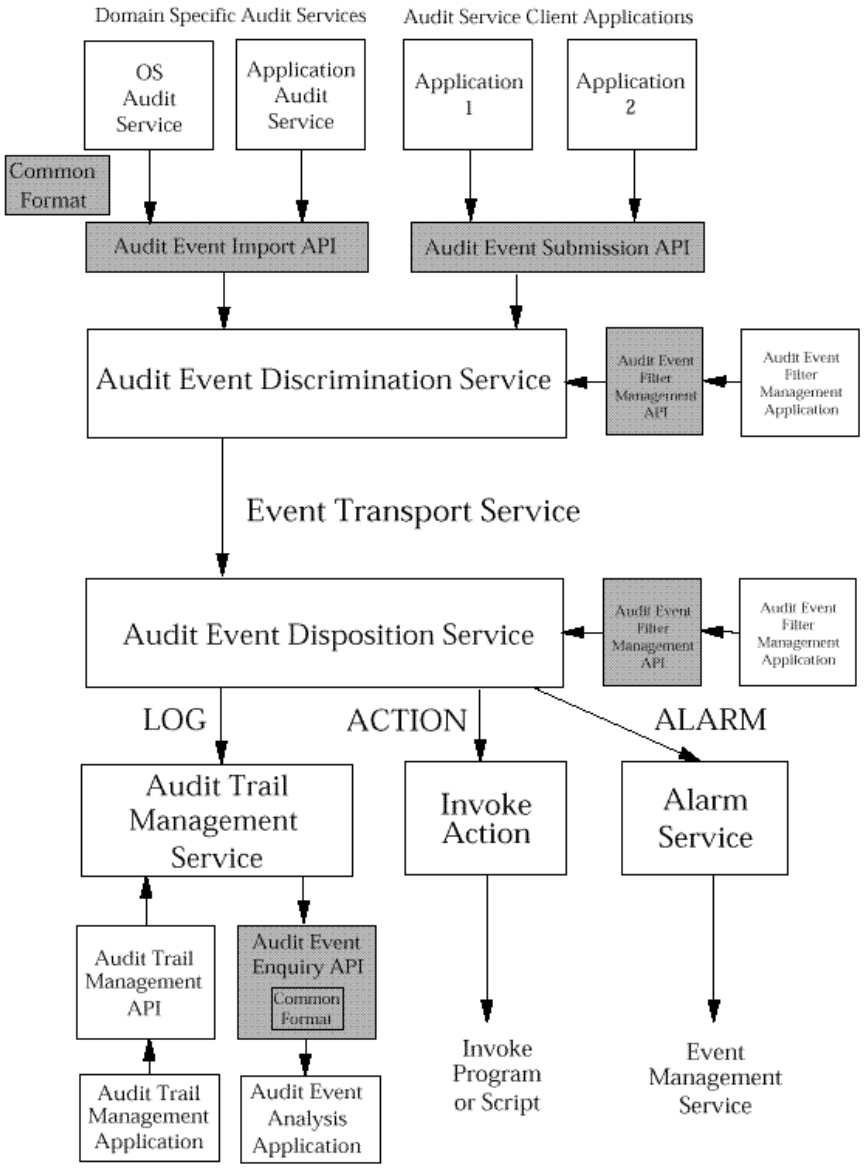


Figure 3-1 Distributed Audit Service Interfaces

In Figure 3-0, the shaded boxes identify the APIs supported by this version of the XDAS specification. The boxes labelled "Common Format" indicate at which APIs the common format defined in this specification is exposed.

The XDAS Audit Service provides an API to support:

- the submission of audit events by applications
- the import of information from audit logs generated by domain specific audit services
- control of the filtering of audit events prior to submission or import
- control of the disposition of events as a combination of any of logging, action initiation and alarm triggering
- the analysis of audit logs.

The Distributed Audit Service model discussed in this section is illustrated in Figure 3-0 (on page 13). This is a logical representation and does not reflect a particular physical architecture. It comprises the following components:

Security Event Detection Service

The *Security Event Detection* service resides in the callers of the XDAS Audit Event Service Client API (shown in the diagram as applications 1 and 2.) An application is responsible for detecting security relevant activity in the context of its own security domain and to generate an audit event record which contains a description of the activity and information about the security context. An application reports the events it detects via the *Audit Event Service Client API.*

Audit Event Import Service

Many domains, in particular operating systems, provide their own audit service designed to meet their domain's specific needs in terms of event types and the information recorded about an event. The *Audit Event Import Service* provides for the import of audit events from a domain specific log for the purposes of merging with XDAS audit information into a time ordered sequence of records for the support of analysis of audit events across domains. In order to use the import service a security domain needs to provide a facility to translate its own audit records into the XDAS common audit event record format.

Note: The translation to the XDAS common audit event record format does not necessarily preserve all information in the original audit record. The XDAS common audit event record format includes information that can be used to locate the original record within the originating domain's audit trail.

Audit Event Discrimination Service

The *Audit Event Discrimination Service* discriminates all incoming events against pre-set criteria which are configured via the *Audit Event Management Service.* Those which do not meet the criteria are ignored. Those which do are passed to the *Audit Event Disposition Service.*

Audit Event Disposition Service

The *Audit Event Disposition Service* receives security relevant events from the *Audit Event Discrimination Service.* Based upon configuration data, the audit disposition service invokes one or more of the following services:

- an *Audit Trail Management Service* for logging the event,
- an *Invoke Action Service* for invoking a command or application configured for invocation on the occurrence of the event.
- an *Alarm Delivery Service* that submits the event to an Event Management Service for handling as a system alarm.

The figure emphasizes that the Event Discrimination Service and Event Disposition Service are not necessarily co-located but may be distributed across different platforms with an

event transport service linking the two components. See Figure 3-2 (on page 16) for more discussion.

Audit Trail Management Service

The *Audit Trail Management Service* receives audit events and stores them in the *Audit Stream*, in an implementation defined format.

The *Audit Trail Management Service* supports:

- The configuration and management of the system resources used to store and process the audit records. For example, files which are often referred to as audit logs. The service allows the location of the audit logs to be defined, as well as how and when the service switches from one audit log to the next in the set. The service also supports the archiving of the audit log in the common audit event record format and the retrieval of logs for analysis

  This version of XDAS is not defining an audit log management API. This is unnecessary for support of the primary objectives of XDAS. XDAS interfaces for recording audit event records and analyzing audit event records perceive the audit log as a single time ordered stream of records.

- The *Audit Trail Enquiry API* provides query access to records on the audit log according to submitted post-selection criteria. The *Audit Trail Enquiry API* presents security audit event information in a common audit log format. See "Common Format" illustrated in Figure 3-0 (on page 13).

3.2 Interfaces

Five application audit APIs are identified in the model but only four are of these are within the current scope of this specification. The four APIs within scope are:

Audit Event Submission API

The *Audit Event Submission API* is defined at the boundary to the *Audit Event Discrimination Service* for submission of audit events detected within application or platform services

Audit Event Import API

The *Audit Event Import API* is defined at the boundary to the *Audit Event discrimination service* for the merging of a set of audit records recorded by a domain specific audit service with the XDAS audit stream. It requires the definition of a common, portable audit log format to support interoperability. See *Common format* in Figure 3-0 (on page 13).

Audit Event Filter Management API

The *Audit Event Filter Management API* is defined to support management applications to configure the Audit Event Discrimination and Audit Event Disposition Services.

Audit Event Enquiry API

The *Audit Event Enquiry API* is defined for the retrieval of audit records in the audit stream.

The fifth API, currently out of the scope of this specification is:

Audit Trail Management API

The *Audit Trail Management API* is defined to configure, manage and archive audit logs that comprise the XDAS audit stream.

3.3 Distributed Audit Service Example

The distributed aspect of an XDAS implementation is illustrated in Figure 3-2. For the purposes of this illustration the XDAS implementation is shown as working over The Open Group Event Management Service. Although this is a possible method of implementation, and one that is capable of supporting interoperability between implementations (to the extent that XEMS supports interoperability) it is not mandated by this specification.

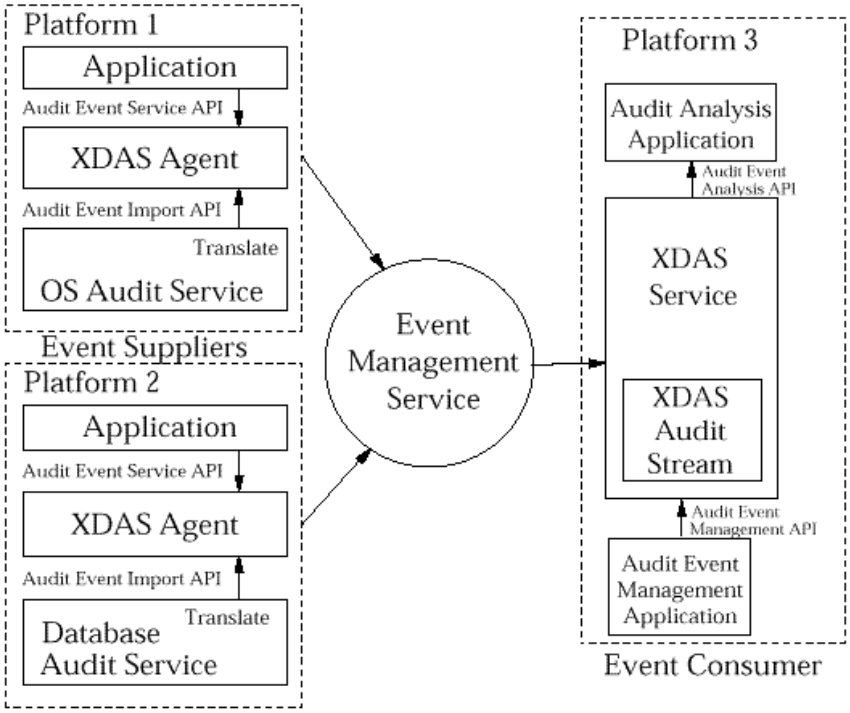


Figure 3-2 Distributed Audit Service Model

3.3.1 XDAS Event Supplier Components

An XDAS component executes on each platform within the distributed system. Those XDAS components providing the **Audit Event Service API** and the **Audit Event Import API** are XEMS Event Suppliers.

Applications may submit audit event records to the XDAS service via the *Audit Event Service API*. Domain specific audit services, such as an operating system audit service, may submit audit event records to the XDAS service for integration with the XDAS Audit Stream. In the case of the *Audit Event Import API* then the caller is required to provide a translation service from the domain specific format to the XDAS common audit event record format.

An XDAS Event Supplier uses the filtering rules to control the events that it submits to the Event Management Service. No decisions regarding the disposition of XDAS events is made by an XDAS Event Supplier.

3.3.2 XDAS Event Consumer Components

The XDAS components that handle the disposition of events are XEMS Event Consumers. The XEMS passes XDAS events submitted to it to XDAS Event Consumers. These components use the action part of the filter rules to control the disposition of the XDAS events received. The actions are to:

- Log the event
- Initiate an action by invoking a program or script
- Initiate an alarm by submitting the XDAS event to the Event Management System as a system alarm.

An audit analysis application is illustrated using the *Audit Event Analysis API* and an Audit Event Management Application using the *Audit Event Management API* from a central XDAS Management platform. The actual location and internal structuring of the XDAS Audit Stream is implementation defined.

The method and format for communicating filtering criteria to the individual XDAS Event Supplier components is not defined by this version of the specification.

Chapter 4

XDAS Data Structures

This chapter presents a definition of the data structures needed for the Distributed Audit Service.

4.1 Audit Record Stream

The XDAS API assumes that audit event records are inserted into and read from a time sequenced stream of audit records in a common format. This stream of records is termed an *Audit Stream*. The organization and management of the system resources used to comprise the audit stream is implementation defined.

4.2 Audit Event Record

Information regarding an audit event is recorded in an *Audit Event Record*. The following section presents a definition of the portable *common exchange format* for audit event records. This is the format in which records are submitted to, or retrieved from, the XDAS API.

The audit record contents are represented using the UTF-8 character encoding. This does not assume that the record contents are in a form that can be displayed as readable text. In addition, manifest constants shall not be localized by any internationalization routines used within XDAS implementations.

The *audit event record* comprises:

- firstly, a minimum set of common information needed to support the filtering of audit events and a top level analysis across the distributed environment for the purposes of traceability and assignment of accountability.
- secondly, for events originated within a domain specific audit service and imported into XDAS, a pointer to the location and position of the original record within the originating domain audit service to support more detailed analysis using domain specific audit tools if required.
- thirdly, provision for recording detailed domain event specific information within the record itself that can be used for more detailed analysis of activity within the context of the service originating the event. When importing records, event specific information may be used instead of or in addition to the pointer to the original record.

Thus, the detailed information from the source domain is not necessarily required for analysis in the context of the distributed environment. For example, an agent may have created objects in a database, the distributed environment may only be interested in the fact that database objects have been created, and not specifically in the type of database object, say a trigger.

In order to be both portable and extensible, the format proposed here adopts an approach based on self-defining attributes expressed in a textual format. See Chapter 6 for the actual format.

The structure of an audit record is as follows:

header
: The header is a mandatory component of an audit event record and contains essential information about the event to be recorded:

- The *length* of the audit record (generated by the implementation).
- The *version_number* of the service, so that analysis tools can accurately interpret the information to follow (generated by the implementation).
- The *date_and_time* of the event (generated by the implementation at the time at which the caller commits an audit event record to the stream or when the function to explicitly timestamp the record is called).

  The XDAS specification includes the date and time specified by reference to the start of the current epoch (0000 GMT 1 January 1970). Time is represented as:

  — The *offset* in millizeconds from the beginning of the epoch

  — The *uncertainty interval* in millizeconds of offset

  — The *uncertainty indicator* as a percentage of confidence in the uncertainty interval

  — The *signal* or *source* of trusted time. This shall typically be the hostname or network address of the network time server.

  — The *timezone* in the format **stdoffset[dst[offset][,start[/time],end[/time]]]**. See Section 6.6 (on page 42).

  The uncertainty interval and uncertainty indicator shall be empty by default. These are considered placeholders for future use.
- The *event_number*, a number which uniquely identifies the event (provided by the caller)
- The *outcome* of the event, i.e., its success or failure (provided by the caller)

originator_information

The originator of an event is defined as the service that detects and requests the recording of an audit event. As such it defines the security domain in which the event occurs.

The *originator_information* is a mandatory component of an audit event record. It is generated by the implementation on the basis of information provided by the caller when an association between the caller and the audit service is initialized.

initiator_information

The initiator of an event is defined as the principal that is accountable for the initiation of the action that results in the audit event.

The *initiator_information* is a mandatory component of an audit event record and is provided by the caller.

target_information

This defines the target on which the initiator has acted. The target may be the identity of a service with which a session has been initiated or terminated.

The *target_information* is an optional component of an audit event record and is provided by the caller.

source_reference

The *source_reference* is an optional component of an audit event record and is provided by the caller. It is a pointer to the original audit event record for those records that have been imported to the XDAS service from a domain specific audit service. The intention is that this information provides the location of the audit record within the original domain if more detailed analysis is required. This information is provided by the original domain when calling the XDAS import API.

event_specific_information

The *event_specific_information* is provided for primary use by applications using XDAS as their primary audit service. *Event_specific_information* varies from one event to the next and is specific to the context of originating security domain identified by the *originator_identity*

The *event_specific_information* is an optional component of an audit record and is provided by the caller. It may include information pertaining to the security context of originator, initiator or target.

The structure of this field is required to be textual, that is, it cannot contain any binary data except in an encoded format. If used, this must be printable UTF-8 character strings consisting of comma separated attribute=value pairs.

4.3 Originator, Initiator and Target Information

4.3.1 Originator_Information

The information associated with an originator, the service that detects and records an audit event, comprises:

- **Location_Name**
  the name of the host/service defined using the syntax and quoting rules defined in Chapter 6.
- **Location_Address**
  this is a communication service end point address. Comparisons on this data should use bitwise comparison.
- **Service_Type**
  the *service_type* may include information about the particular subset of functions being provided by the originator. For example, a service provider may support different subsets of functions according to the port by which it is invoked. It is represented as a text string.
- **Originator Authentication Authority**
  is defined using the syntax and quoting rules defined in Chapter 6. Examples of an *authentication authority* are the name of a kerberos realm, an NIS domain, and a UNIX hostname.
- **Originator Name**
  the originator principal name as authenticated by authentication authority. Examples of principal names are a kerberos principal name, and a UNIX username.
- **Originator Identity**
  the originator principal identity. Examples are the DCE UUID and a UNIX uid.

It is not mandatory that both the *location_name* and the *location_address* are completed, but at least one of them shall be.

The *originator authentication authority*, *originator name* and *originator identity* represent the authenticated identity of the originator. The *originator authentication authority* and *originator identity* are mandatory information. The *originator name* is optional and may be left empty.

4.3.2 Initiator_Information

The information associated with an initiator comprises

- **Initiator Authentication Authority**
  defined using the syntax defined in Chapter 6. Examples of an *authentication authority* are the name of a kerberos realm, an NIS domain, and a UNIX hostname.
- **Initiator Name**
  the initiator principal name. Examples of principal names are a kerberos principal name, and a UNIX username.
- **Initiator Identity**
  the initiator principal identity. Examples of principal identities are a DCE UUID and a UNIX uid.

Note: It should be noted that in some circumstances it may be desirable or required by regulation that events are not associated directly with individual users without an additional reference stage in the analysis. This may influence the information that is actually stored in an XDAS record. For example, an audit ID distinct from an access ID or principal ID may be used within an audit event record.

The *initiator authentication authority* and *initiator identity* are mandatory information. The *initiator name* is optional and may be left empty.

4.3.3 Target_Information

The target of an activity that results in an auditable event may be:

- an "object" that may be identified by a name within the originating domain's namespace. For example a file on a UNIX platform, a record within a database.
- a service with which an association is established.

  In the case of client-server operations, when an association is created then both ends may be considered to be the target of the other even though strictly speaking one side is the initiator. For events recording the creation of associations the target_information therefore records information about the remote service component. The initiator_information therefore always references the original (normally end-user) principal.

  The service may assign its own representation of the principal identity to the client (e.g., using a local account database). In this case the identity assigned needs to be recorded to support traceability at the distributed system level.

Not all events necessarily have a target. When a target is relevant to an event the *target authentication authority*, *target name* and *target identity* represent the authenticated identity of the target. The *target authentication authority* and *target identity* are mandatory information if a target is recorded. The *target name* is optional and may be left empty.

4.4 Identification of Audit Events

The identification of audit events is an important part of supporting requirements to filter and select audit events.

Audit Events may be specifically referred to by an *Event Number*. A set of Audit Events may be referred to by an *Event Class*. An XDAS defined set of generic Event Classes are listed at the end of this section.

The purpose of defining *Event Classes* is to facilitate the definition of filtering criteria for the control of the audit service and for facilitating the definition of search criteria for audit analysis. An audit event record only includes the *Event Number*. It does not include any reference to *Event Class*

4.4.1 Event Numbers

XDAS defines a minimum required set of generic event numbers. It is possible for application developers to register their own additional event numbers if they wish to utilize the services of XDAS for more domain specific auditing not catered for by the generic set of XDAS events.

A conforming implementation is required to treat as valid all the XDAS audit events and event classes defined in this specification and to provide an implementation defined method for configuring additional registered event numbers and event classes as valid.

4.4.2 XDAS Events

The following generic events are registered by XDAS. Not all of these events are necessarily security significant within all domains. For example the querying of attributes or configuration data is not necessarily of security significance.

Account Management Events

This set of events is applicable to the management of principal accounts. A principal may be an end-user or a service within the system, a psuedo-user.

- **Create account**
  The creation of an account representing a principal within a domain.
- **Delete account**
  The deletion of an account representing a principal from a domain.
- **Disable account**
  An action the prevents a principal account from being used within a domain.
- **Enable account**
  An action that permits a principal account to be used within a domain.
- **Query account attributes**
  The requesting of the attributes associated with a principal within a domain.
- **Modify account attributes**
  The modification of the attributes associated with a principal within a domain.

User Session Events

This set of events is relevant to the creation and use of user sessions on the system.

- **Create a user session**
  The establishment of a processing environment to service an end user.
- **Terminate a user session**
  The dismantling of a processing environment associated with servicing an end user.

- **Query user session attributes**
  The requesting of the attributes associated with a user session.
- **Modify user session attributes**
  The modification of security significant attributes of the context of a processing environment servicing an end user.

Data item and Resource Element Management Events

This set of events relate to the creation and management of data items and resource elements within a domain. The type of data item or resource element is dependent upon the domain, e.g., files and directories, device special files, shared memory segments, within an operating system, tables and records within a database, messages within an email system. The term data item is used to refer to any type of resource element.

- **Create data item**
  Creation of a data item within a domain.
- **Delete data item**
  Deletion of a data item from a domain.
- **Query data item attributes**
  The requesting of the attributes associated with a domain data item.
- **Modify data item attributes**
  The modification of the security attributes of a domain data item such as access control attributes, ownership, aliases.

Service or Application Management Events

This set of events relate to the management of system services and applications.

- **Install service or application**
  The installation of additional or updated software on a system, e.g., an application or system service.
- **Remove service or application**
  The deinstallation of software on a system.
- **Configure service or application**
  The modification of the configuration data associated with a software component.
- **Query configuration of service or application**
  The requesting of information about the configuration of a service or application.
- **Disable service or application**
  An action that prevents an application or system service from being used, for example, inhibiting responses to service requests. It may also involve the termination (shutdown) of application processing components that are currently providing the service.
- **Enable service or application**
  An action that permits an application or system service to be used, for example, allowing responses to service requests. This may also involve the invocation of specific application processing components (startup).

Service and Application Utilization Events

These events relate to the use of service and applications. They typically map to the execution of a program or a procedure and manipulation of the processing environment.

- **Invoke service or application**
  The invocation of a service or application (exec), e.g., operating system utility, database, accounting application, etc.

- **Terminate service or application component**
  The termination (exit) of the use of a service or application. This could be at the instigation of the application itself or by the intervention of the domain in response to user or administrative action.
- **Query processing context**
  The requesting of the attributes associated with the current processing environment.
- **Modify processing context**
  The modification of the attributes associated with the current processing environment.

Peer Association Management Events

- **Create an association with a peer**
  The creation of a communication channel and the processing context between system components.
- **Terminate an association with a peer**
  The closure of a communications channel and destruction of processing context between system components.
- **Query an association context**
  The requesting of the attributes of a context associated with a communications channel between peers.
- **Modify an association context**
  The modification of the attributes of a processing context associated with a communications channel.
- **Receive data via an association**
  Receiving data from associated peer within current association context.
- **Send data via an association**
  Sending data to associated peer within current association context.

Data Item or Resource Element Content Access Events

These events relate to the formation of an association between a service or application and a data item or resource element for the purpose of using its contents or services. For example, a file or directory, device special file, memory segment, communications port, etc.

- **Create association with data item**
  Create an association with (open) a data item. This creates a binding between the caller and the data item.
- **Terminate association with data item**
  The termination of an existing association with (close) a data item.
- **Query context of association with data item**
  The requesting of the context of an association with a data item, e.g., mode of access, size limits, access path, etc.
- **Modify context of association with a data item**
  The modification of the context of an association with a data item or resource element.
- **Query data item contents**
  The requesting of the contents of a domain data item (read).
- **Modify data item contents**
  The modification of the contents of a domain data item (write, append, etc.).

Exceptional Events

These are events that are considered to be outside the generalized events listed above.

- **Start system**
  The action of booting a system host or of changing the processing state of a system host to an operational mode.
- **Shutdown System**
  The action of halting the processing by a system host or of changing the processing state of a system host to a maintenance mode.
- **Resource exhaustion**
  The detection of resource exhaustion which has a potential impact on system operations, perhaps based upon a configurable threshold, e.g., data storage resources, communication end points.
- **Resource corruption**
  The detection of an integrity failure of a system resource, for example data storage resource.
- **Backup datastore**
  The action of making a backup copy of a datastore for the purposes of protecting availability and integrity of the data it contains.
- **Recover datastore**
  The action of restoring the contents of a datastore from a previously made backup copy for the purposes of restoring the availability of the contents, or the integrity of the contents, or both.

Audit Service Management Events

These are events of specific relevance to the audit service itself.

- **Configure audit service**
  The modification of the parameters controlling the operation of the audit service, for example, audit event filtering criteria.
- **Audit datastore full**
  The detection of resource exhaustion for the particular instance of the resource used to store the log of audit event records.
- **Audit datastore corrupted**
  The detection of a datastore integrity failure for the particular instance of the resource used to store the log of audit event records.

4.4.3 Event Classes

Audit Events may be specifically referenced by an *Event Number*. A set of Audit Events may be referred to by an *Event Class*. The concept of an *Event Class* is included in the XDAS solely as an administrative convenience. It provides an efficient and convenient reference to sets of audit events so that audit filters can be easily defined. An audit event record only includes the *Event Number*. It does not include any reference to *Event Class* for two reasons: its inclusion leads to redundant information in the audit record; and the mapping of event classes across administrative domains is problematic. When specified in filtering selection criteria, an *event class* is translated internally into the individual event numbers.

Default Event Classes

The XDAS defines a default set of event classes. Others can be defined by the implementation and configured by a system administrator to group together XDAS event numbers in a meaningful way. The default set of event classes defined by the XDAS are listed below:

- Account management events
- User session events
- Data item and resource element management events
- Service and application management events
- Peer association management
- Data item or resource element content access events
- Exceptional events
- Audit service management events

The default mapping of events to these event classes is as listed in Section 4.4.2.

4.4.4 Outcomes

An event shall be identified by both its event number and outcome. The following outcome codes and sub-codes are defined by this specification:

Outcome	Outcome Description
Successful	XDAS_OUT_SUCCESS XDAS_OUT_PRIV_USED XDAS_OUT_PRIV_GRANTED XDAS_OUT_PRIV_REVOKED XDAS_OUT_PRESELECT_CRITERIA_SET XDAS_OUT_THRESHOLDS_SET XDAS_OUT_ACTIONS_SET XDAS_OUT_THRESHOLD_EXCEEDED
Failure	XDAS_OUT_FAILURE XDAS_OUT_SERVICE_UNAVAILABLE XDAS_OUT_SERVICE_FAILURE XDAS_OUT_HARDWARE_FAILURE XDAS_OUT_LOST_ASSOCIATION XDAS_OUT_ALREADY_ENABLED XDAS_OUT_ALREADY_DISABLED XDAS_OUT_SERVICE_ERROR XDAS_OUT_BUSY XDAS_OUT_DISABLED XDAS_OUT_INVALID_INPUT XDAS_OUT_ENTITY_EXISTS XDAS_OUT_ENTITY_NON-EXISTENT
Denial	XDAS_OUT_DENIAL

Outcome	Outcome Description
	XDAS_OUT_INSUFFICIENT_AUTHORIZATION
	XDAS_OUT_INVALID_IDENTITY
	XDAS_OUT_INVALID_CREDENTIALS

4.5 Event Selection

Field	Event Submission	Event Import
Header:		
Length	-	-
Version	X	X
Date	-	X
Event Number	X	X
Outcome	X	X
Originator_Information:		
location_name	-	X
location_address	-	X
service_type	-	X
auth_authority	-	X
name	-	X
identity	-	X
Initiator_information:		
auth_authority	X	X
name	-	X
identity	-	X
Target_information:		
location_name	-	X
location_address	-	X
service_type	-	X
auth_authority	-	X
name	-	X
identity	-	X
Source:	-	-
Event_Specific:	-	-

Table 4-1 Event Filtering Criteria

Event selection criteria may be applied at the two places within the XDAS architecture at which events are entered into the service:

- Preselection criteria may be applied when an event is submitted to determine whether an event is to be audited.
- Selection criteria may be applied when an event is imported from a domain specific audit service to determine whether the event is to be imported.

Table 4-1 sets out the preselection filtering criteria. An "X" indicates that the field is available for filtering; a "-" that it is not.

4.5.1 Event Submission Preselection Filtering

The filtering criteria for preselection of events on event submission is constrained by considerations of limiting the performance impact of evaluating the criteria on the calling application and the system as a whole.

Whilst date and time of day are valid requirements for filtering on event submission, they are not included as mandatory requirements in the table. This is because this selection can be achieved more efficiently using a scheduling service to switch event filtering criteria as a whole.

The event originator is not included in the table, even though it is a valid requirement for filtering. This filtering can be achieved more easily as an application level facility which turns auditing on or off for the application as a whole, or for subservices within an application. It is not considered to be a valid XDAS function.

Filtering by initiator *auth_authority* is a requirement as an *auth_authority* may be compromised or otherwise untrusted. However, controlling filtering by individual identity impacts performance significantly and thus, it is not a mandatory requirement in the XDAS. Such filtering is more efficiently performed on import or post-selection analysis.

4.5.2 Event Import Preselection Filtering

XDAS supports a much richer set of filter criteria for controlling the selection of records for import to XDAS as the performance impact is of lesser concern in this case.

4.5.3 Event Filters

An event filter comprises the following information:

Version Number
The XDAS version number.

Filter Name
A name by which the filter is referred to.

Filter Type
The filter applies to the event submission or event import interface, or both interfaces.

Flag
The flag which indicates whether the filter is enabled or disabled.

Expression List
A set of expressions which must all be satisfied to establish the complete filter to be applied.

Action List
The actions to be taken when the event is detected.

4.5.4 Filter Expression List

A filter expression list comprises a set of expressions which must all be satisfied to establish the complete expression to be applied. This specification does not assume any precedence or ordering of the evaluation of a set of filters (although an implementation may apply one for performance reasons). If an event requires auditing under the filtering criteria of any individual filter then it shall be audited, even if excluded by other filters. In the circumstance that an event is required to be audited by multiple filters then duplicate audit event records shall not be created.

An expression comprises:

Include/Exclude Flag
Events matching this expression are to be included or excluded from selection. When a filter is evaluated all inclusions are processed first and followed by all exclusions.

Attribute
The event attribute or field.

Operator
The operator defines the boolean operation to be performed on the attribute. Operators are equal, greater than, less than, greater than or equal, less than or equal, not equal, test for bits set, substring.

Note: The event outcome codes have been structured for convenience in checking using a test for bits set.

Value
The value against which the attribute value in the event is tested.

4.5.5 Filter Action List

The filter action list comprises a sequence of action definitions. Each action definition comprises an action bit mask to indicate the action(s) to be taken and a text string that is used to further define the action(s).

Action Mask
The action(s) to be taken. This can be LOG, ALARM or ACTION or any combination.

Text String
A text string that provides additional information pertinent to the action to be taken. The format of this string is implementation defined.

Examples of the *filter action list* are

- LOG + Empty string
- ALARM + Severity Code
- ACTION + Pathname of executable or script to invoke and input parameters

Chapter 5

DAS Usage Model

The XDAS comprises both operational and management services. The operational XDAS services are those available to applications in support of the logging of audit records. The management services support the configuration and management of audit events, the audit service itself, as well as providing interfaces for the analysis of audit records.

The XDAS places a dependency on an Event Management Service such that the intermediate event management components do not modify the filtering or routing of audit events, thereby ensuring that an audit alarm, for example, is not filtered out part way to its destination

Operational services include:

- *General Audit Service API*, used by all callers of the XDAS.

  All callers are required to initiate a session with the XDAS audit service. This authenticates the caller's identity and establishes a session between the caller and the XDAS. Thereafter, callers may use the XDAS APIs to log events, configure the audit service, or analyze audit streams subject to the XDAS authorities assigned to them.

- The *Audit Event Service Client API*, used by applications to submit security relevant events to the Audit Service.

  These allow audit records to be created, filled and committed to the implementation defined audit log in common format.

- The *Audit Log Import API*, used by domain specific audit services to import audit records in the XDAS common audit event record format int the XDAS audit stream.

Management services include:

- The *Audit Event Management API*, used by applications to configure the preselection criteria for the *Audit Event Discrimination Service* and the *Audit Event Disposition Service*

- The *Audit Read API*, used by applications to retrieve events from the audit stream for the purposes of analysis.

5.1 Authorization Policy

The authorization policy inherent in the XDAS-API is defined on the principle of the separation of duties. The granting of XDAS authorities is under the control of authorization security services. The following XDAS authorities have been defined:

XDAS_AUDIT_SERVICE
: required to initialize a session with the XDAS audit service.

XDAS_AUDIT_SUBMIT
: for using the audit logging interfaces of the Audit Event Service Client.

XDAS_AUDIT_IMPORT
: required to import audit events records from a domain specific audit service.

XDAS_AUDIT_CONTROL
: for use of the Audit Event Management APIs.

XDAS_AUDIT_READ
for access to the Audit Read API.

Each interface specification includes the XDAS authority required to be possessed by a caller in order to utilize the interface. The mechanism for enforcement of the authorization policy is implementation specific. Support is included in this specification for the initialization of a session between a caller and the XDAS service whereby the identity of the caller can be authenticated and appropriate authorization attributes established.

5.2 General Audit Service API

Initialize Session
Initialize a session with the XDAS. This call will fail unless the caller possesses at least one XDAS authority.

Terminate Session
Terminate a session with the XDAS

All callers must initiate a session with the XDAS before they can use any of the services it provides. The initialization of the session supports the mutual authentication of the audit client and audit service components and establishes the audit client's XDAS authorities The caller is returned a handle to the XDAS service which is then used for all XDAS API functions. On completion, the caller must terminate the XDAS session.

The behaviour if a client dies or exits without calling *terminate session* is implementation defined. An implementation may take specific action to try and detect and terminate such sessions itself to address any potential denial of service risks.

5.3 Audit Event Service Client API

Start Record
Allocate and initialize an audit record descriptor. The return from this indicates to the caller whether the event requires auditing or not under the current filtering criteria.

Put Event Information
Add event specific information to the initialized audit record

Commit Record
Write the audit record to the audit log

Discard Record
Discard the audit record

Time Stamp Record
Control the time at which the record is timestamped

Callers submit security relevant events to the *Audit Event Service Client API.* The functions build the record from the information given by the caller and from the processing environment. The interfaces cover the creation, filling and committing of an audit record to the audit trail.

5.4 Audit Log Import API

Import_Event_Records
This function supports the import to XDAS by another audit service of multiple audit event records formatted in the XDAS common audit event record format.

This service permits domain specific audit services to import their own audit records into the XDAS service for consolidation and analysis at the distributed system level. Only callers with the XDAS_AUDIT_IMPORT authority are permitted to use this function.

5.5 Audit Event Management API

Create Filter
Create or modify an audit filter defining the selection criteria and the action to be taken on detection.

List Filters
Get a list of the names of filters which have been defined

Release Filter List
Release the list of filter names returned by List Filters

Get Filter
Get the specified audit filter

Delete Filter
Delete the specified audit filter

Enable Filter
Enable the specified filter

Disable Filter
Disable the specified filter

The *Audit Event Management API* provides the means whereby the *Audit Event Discrimination Service* and the *Audit Event Disposition Service* are configured. Only callers with the XDAS_AUDIT_CONTROL authority are permitted to use these interfaces.

5.6 Audit Read API

Open Audit Stream
Open the XDAS audit stream for read

Rewind Audit Stream
Rewind the audit stream

Close Audit Stream
Close the XDAS audit stream

Get Next
Read the next set of audit records from the specified audit trail into buffer. The caller supplies the buffer length and the maximum number of records to be returned. The implementation may return as many records as will fit into the buffer up to the specified

maximum. The caller can then parse the buffer to extract individual records.

The *Audit Read API* is used to extract records from the XDAS audit stream for analysis. The interface supports the copying of a record into a buffer where the contents may be examined by the caller. The interfaces are available to privileged callers who possess the XDAS_AUDIT_READ authority.

Chapter 6

Parameter Passing Conventions

This chapter describes the data types and constants used by the the XDAS functions. It also explains calling conventions for these functions.

6.1 Structured Data Types

Wherever these XDAS-API C-bindings describe structured data, only fields that must be provided by all XDAS-API implementations are documented. Individual implementations may provide additional fields, either for internal use within XDAS-API routines, or for use by non-portable applications.

6.2 Integer Types

XDAS-API defines the following integer data type

```
OM_uint32     32-bit unsigned integer
```

Where guaranteed minimum bit-count is important, this portable data type is used by the XDAS-API routine definitions. Individual XDAS-API implementations include appropriate **typedef** definitions to map this type onto a built-in data type.

6.3 String Data and Similar Data

6.3.1 Byte Strings

Many of the XDAS-API routines take arguments and return values that describe contiguous multi-byte data. All such data are passed between the XDAS-API and the caller using the **xdas_buffer_t** data type. This data type is a pointer to a buffer descriptor consisting of a **length** field, which contains the total number of bytes in the data, and a **value** field, which contains a pointer to the actual data:

```
typedef struct xdas_buffer_desc_struct{
   size_t   length;
   void    *value;
} xdas_buffer_desc, *xdas_buffer_t;
```

Storage for data passed to the application by a XDAS-API routine using the **xdas_buffer_t** conventions is allocated by the XDAS-API routine. The application may free this storage by invoking the *xdas_release_buffer*( ) routine. Unused **xdas_buffer_desc** objects shall have length set to 0, and value set to NULL.

6.3.2 Character Strings

Certain multi-octet data items may be regarded as UTF-8 character strings as defined in Internet RFC 2044. Character strings are passed between the application and the XDAS-API using the **xdas_buffer_t** data type, defined earlier.

6.3.3 Opaque

Certain multi-octet data items are considered opaque data types at the XDAS-API, because their internal structure only has significance to the implementation. Examples of such opaque data types are

audit service handle
: This is opaque to the caller and returned to the caller on initialization of a session between the caller and the XDAS audit service. It is subsequently passed as a parameter to each XDAS-API call as a **xdas_audit_ref_t** data type.

audit stream handle
: This is opaque to the caller and is returned to a caller of the *xdas_open_audit_stream*( ) function. It is subsequently passed as a parameter to those functions that manipulate an audit stream as a **xdas_audit_stream_t** data type.

audit record descriptor
: This is opaque to the caller and is returned to a caller of the *xdas_start_record*( ) function. It is subsequently passed as a parameter to those functions that manipulate an audit record for submission to the XDAS service as a **xdas_audit_rec_desc_t** data type.

6.4 XDAS Audit Event Record Format

The audit event record format is defined as an UTF-8 character encoding in an **xdas_buffer_t** structure. Fields are delineated with colons (:); where a colon is part of the alphanumeric string. ‘‘%’’ shall be used as the escape character. The character immediately following a ‘‘%’’ is not interpreted. For example, ‘‘%%’’ yields ‘‘%’’, ‘‘%:’’ yields ‘‘:’’, ‘‘%%%:’’ yields ‘‘%:’’. Empty strings are represented by two adjacent separator characters. Note that this is an ordered sequence. The sequence of fields in the XDAS audit event record format is set out below:

Field	Type
Header:	"HDR"
<length_in_bytes>	Digits 0-9
<version>	Digits 0-9
<time_offset>	Hexadecimal
<time_uncertainty_interval>	Hexadecimal
<time_uncertainty_indicator>	Hexadecimal
<time_source>	Alphanumeric
<time_zone>	Alphanumeric
<event_number>:	Hexadecimal
<outcome>	Hexadecimal
Originator	"ORG"
<org_location_name>	Alphanumeric
<org_location_address>	Alphanumeric
<org_service-type>	Alphanumeric
<org_auth_authority>	Alphanumeric
<org_principal_name>	Alphanumeric
<org_principal_id>	Alphanumeric
Initiator	"INT"
<int_auth_authority>	Alphanumeric
<int_domain_specific_name>	Alphanumeric
<int_domain_specific_id>	Alphanumeric
Target	"TGT"
<tgt_location_name>	Alphanumeric
<tgt_location_address>	Alphanumeric
<tgt_service-type>	Alphanumeric
<tgt_auth_authority>	Alphanumeric
<tgt_principal_name>	Alphanumeric
<tgt_principal_id>	Alphanumeric
Source	"SRC"
<pointer_to_source_domain>	Alphanumeric
Event	"EVT"
<event_specific_information>	Alphanumeric
End	"END"

The strings HDR, ORG, INT, TGT, SRC and EVT are included to support syntax checking. All fields shall be included in the audit record, with separators, even if they are empty (e.g., ::).

The following structure is defined for returning an audit event record

```
typedef struct xdas_audit_record_desc_struct{
    const OM_uint32               record_number,
    OM_uint32                     length,
    OM_uint32                     version,
    OM_uint32                     time_offset,
    OM_uint32                     time_uncertainty_interval,
    OM_unit32                     time_uncertainty_indicator,
    xdas_buffer_t                *time_source,
    xdas_buffer_t                *time_zone,
    OM_unit32                     event_number,
    OM_unit32                     outcome,
    xdas_buffer_t                *org_location_name,
    xdas_buffer_t                *org_location_address,
    xdas_buffer_t                *org_service_type,
    xdas_buffer_t                *org_auth_authority,
    xdas_buffer_t                *org_principal_name,
    xdas_buffer_t                *org_principal_identity,
    xdas_buffer_t                *int_auth_authority,
    xdas_buffer_t                *int_principal_name,
    xdas_buffer_t                *int_principal_identity,
    xdas_buffer_t                *tgt_location_name,
    xdas_buffer_t                *tgt_location_address,
    xdas_buffer_t                *tgt_service_type,
    xdas_buffer_t                *tgt_auth_authority,
    xdas_buffer_t                *tgt_principal_name,
    xdas_buffer_t                *tgt_principal_identity,
    xdas_buffer_t                *source_reference,
    xdas_buffer_t                *event_info
} xdas_audit_record_desc, *xdas_audit_record_t;
```

The components of the audit record structure are:

length
The length in bytes of the retrieved record.

version
The version number of the XDAS service that created the audit record.

time_offset
The time at which the audit record was committed or was timestamped by a specific function call.

time_uncertainty_interval
The interval of time by which the time recorded for this event is uncertain.

time_uncertainty_indicator
The percentage of confidence in the time_uncertainty interval.

time_source
The name or address of the source of the time recorded for this event.

time_zone
The time zone applicable to the domain in which the event occured.

event_number
The event number defining the type of event.

outcome
The outcome code recorded for the event.

org_location_name
The name of the location of the originator domain.

org_location_address
The address of the location of the originator domain.

org_service_type
The service type of the originator domain.

org_auth_authority
The name of the authentication authority for the originator principal.

org_principal_name
The name of the originator principal.

org_principal_identity
The identity of the originator principal.

int_auth_authority
The name of the authentication authority for the initiator principal.

int_principal_name
The name of the initiator principal.

int_principal_identity
The identity of the initiator principal.

tgt_location_name
The name of the location of the target domain.

tgt_location_address
The address of the location of the target domain.

tgt_service_type
The service type of the target domain.

tgt_auth_authority
The name of the authentication authority for the target principal.

tgt_principal_name
The name of the target principal.

tgt_principal_identity
The identity of the target principal.

source_reference
For an imported record, the pointer to the original record within the originating domain.

event_info
The event specific information recorded for the record.

6.5 XDAS Name String Syntax

An *XDAS name* used within an XDAS audit event record consists of an ordered list of zero or more components. This is termed a composite name. Each component is a string name from the namespace of a single naming system and uses the naming syntax of that naming system. A component may be an atomic or a compound name from that namespace.

A composite name is the concatenation of the components of the name from left to right with the *XDAS component separator* character ('/') separating each component.

6.5.1 Encoding of XDAS Composite Name Strings

Special characters used in the XDAS composite name syntax, such as the component separator or escape characters, have the same encoding as they would in UTF-8

The minimum requirement for all XDAS implementations is to support UTF-8 for communication of name strings.

6.5.2 Backus-Naur Form (BNF) of XDAS Composite Names

This section defines the standard string form of XDAS composite names in BNF. Note that all the characters of the string representation of one name must uniformly use the same encoding and locale information.

The notations used are as follows:

Symbol	Meaning
::=	Is defined to be
|	Alternatively
<text>	Non-terminal element
""	Literal expression
*	The preceding syntactic unit can appear 0 or more times.
+	The preceding syntactic unit can appear 1 or more times.
{}	The enclosed syntactic units are grouped as a single syntactic unit (can be nested).

The XDAS composite name syntax in BNF is as follows.

```
NULL ::=            // Empty set

<PCS> ::=           // Portable Character Set
                        // The set consists of the glyphs:
                        //   !"#$%&'()*+,-./0123456789:;<=>?
                        //  @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
                        //  `abcdefghijklmnopqrstuvwxyz{|}~

<CharSet> ::=       <PCS>
                    | Characters from the repertoire of a string representation

<EscapeChar> ::=    %

<ComponentSep> ::=  /

<Quote1> ::=        "

<Quote2> ::=        ´
```

```
<MetaChar> ::=      <EscapeChar> | <ComponentSep>

<SimpleChar> ::=    // any character from <CharSet> with <ComponentSep>, <Quote1>,
                    // and <Quote2> excluded. An <EscapeChar> <MetaChar>, or
                    // <EscapeChar> <Quote1>, or <EscapeChar> <Quote2> is
                    // substituted by the corresponding unescaped character and
                    // is equivalent to a <SimpleChar>.

<Component> ::=     <SimpleChar>*
                    | <SimpleChar>+ {<Quote1> | <Quote2> | <SimpleChar>}*
                    |   <Quote1>   <CharSet>*   {<EscapeChar><Quote1>}*   <CharSet>*
                    <Quote1>
                            // <CharSet> must not contain unescaped <Quote1>
                            // (note that <Quote2> can appear unescaped)
                    |   <Quote2>   <CharSet>*   {<EscapeChar><Quote2>}*   <CharSet>*
                    <Quote2>
                            // <CharSet> must not contain unescaped <Quote2>
                            // (note that <Quote1> can appear unescaped)

<CompositeName> ::= NULL
                    | <Component> {<ComponentSep> <Component>}*
```

6.6 Time Zone Field

The format of the *time_zone"* field in the XDAS audit record is as defined in the Single UNIX Specification. It is repeated here for information.

```
std offset [dst [offset] [rule]]
```

Where:

- **std and dst**
  Indicates no less than three, nor more than {TZNAME_MAX}, characters that are the designation for the standard (std) or the alternative (dst - such as Daylight Savings Time) timezone. Only std is required; if dst is missing, then the alternative time does not apply in this locale. Upper- and lower-case letters are explicitly allowed. Any graphic characters except a leading colon (:) or digits, the comma (,), the minus (–), the plus (+), and the null character are permitted to appear in these fields, but their meaning is unspecified.
- **offset**
  Indicates the value one must add to the local time to arrive at Coordinated Universal Time. The offset has the form:

  ```
  hh[:mm[:ss]]
  ```

  The minutes (mm) and seconds (ss) are optional. The hour (hh) is required and may be a single digit. The offset following std is required. If no offset follows dst, the alternative time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour is between zero and 24, and the minutes (and seconds) if present between zero and 59. Use of values outside these ranges causes undefined behaviour. If preceded by a –, the timezone is east of the Prime Meridian; otherwise it is west (which may be indicated by an optional preceding +).
- **rule**
  Indicates when to change to and back from the alternative time. The rule has the form:

  ```
  start_date[/time], end_date[/time]
  ```

  where the first date describes when the change from standard to alternative time occurs and the second date describes when the change back happens. Each time field describes when, in current local time, the change to the other time is made.

  The format of date is one of the following:

- **Jn**
  The Julian day n (1 n 365). Leap days are not counted. That is, in all years including leap years February 28 is day 59 and March 1 is day 60. It is impossible to refer explicitly to the occasional February 29.
- **n**
  The zero-based Julian day (0 n 365). Leap days are counted, and it is possible to refer to February 29.
- **Mm.n.d**
  The dth day (0 d 6) of week n of month m of the year (1 n 5, 1 m 12, where week 5 means "the

last d day in month m" which may occur in either the fourth or the fifth week). Week 1 is the first week in which the day occurs. Day zero is Sunday.

The time has the same format as offset except that no leading sign (- or +) is allowed. The default, if time is not given, is 02:00:00.

Note: Where a ':' character is used within the timezone definition then it will need to be escaped by the '%' character when inserted into the timezone field in an audit record.

6.7 Filters

XDAS audit event filters are created using the function xdas_create_filter( ).

6.7.1 Filter Expressions

Filters are used to set the criteria for preselecting events to be recorded, or for selecting records to be imported from an audit stream.

A filter expression is defined as a **xdas_buffer_t** data type containing UTF-8 characters. It is a sequence of variable length fields, separated by a ":" delimiter, as set out below. Note that if a colon is part of an alphanumeric string, then it shall be escaped. (See Section 6.4 (on page 36).) The format for a single filter expression is set out below:

Field
Include / Exclude Flag
Attribute
Operator
Value

The Include / Exclude flag, Attribute and Operator fields are the manifest names defined in Table 6-9 (on page 55), Table 6-10 (on page 55) and Table 6-11 (on page 56). The value field is an unspecified string.

A filter may be defined with a list of filter expressions which shall be evaluated in the sequence in which they are listed. The intention is that a subsequent expression may define exceptions to a previous expression. Thus an expression may exclude a set of events based on event class, but a subsequent expression, based on event number, may specifically include a subset of the events otherwise excluded by reference to event class.

6.7.2 Filter Actions

A filter may also define the disposition of an event submitted to, or imported to the XDAS service.

A filter action is defined as a **xdas_buffer_t** data type containing UTF-8 characters. It is a sequence of variable length fields, separated by a ":" delimiter, as set out below. Note that if a colon is part of an alphanumeric string, then it shall be escaped. (See Section 6.4 (on page 36).) The format for a single filter expression is set out below:

Field
Action Mask
Text String

The components of the action mask are the manifest names defined in Section 6.11.5 (on page 56). The format of the text string is implementation defined.

A filter may be defined with a list of filter actions which shall be executed in the sequence in which they are listed.

6.8 Status Values

One or more status codes are returned by each XDAS-API routine. Two distinct sorts of status code are returned. These are termed XDAS status codes and minor status codes. An implementation of XDAS functions shall return [XDAS_S_COMPLETE] and other status values appropriate for the implementation of the function. The characteristics of a particular implementation may make some status returns inappropriate for that implementation.

6.8.1 XDAS Status Codes

XDAS-API routines return XDAS status codes as their **OM_uint32** function value. These codes indicate major status errors that are independent of the underlying mechanism used to provide the security service.

A XDAS status code can indicate a single fatal generic API error from the **routine error** and a single **calling error**. These errors are encoded into the 32-bit XDAS status code as illustrated in Figure 6-1.

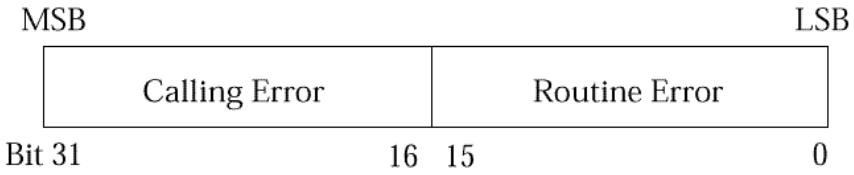


Figure 6-1 Status Code

Hence if a XDAS-API routine returns a XDAS status code containing a non-zero value, the call failed. If the **Calling Error** field is non-zero, the invoking application's call of the routine was erroneous. Calling errors are defined in Table 6-1. If the **Routine Error** field is non-zero, the routine failed for one of the routine-specific reasons listed in Table 6-2 (on page 47).

Name	Value in Field	Meaning
[XDAS_S_CALL_INACCESSIBLE_READ]	1	A required input argument cannot be read.
[XDAS_S_CALL_INACCESSIBLE_WRITE]	2	A required output argument cannot be written.
[XDAS_S_CALL_BAD_STRUCTURE]	3	An argument is malformed.

Table 6-1 Calling Errors

Name	Value in Field	Meaning
[XDAS_S_COMPLETE]	0	Successful completion.
[XDAS_S_AUTHORIZATION_FAILURE]	1	The caller does not possess the required authority.
[XDAS_S_BUFF_TOO_SMALL]	2	The buffer allocated by the caller is too small to hold a single audit record.
[XDAS_S_END]	3	The end of the audit stream has been reached.
[XDAS_S_FAILURE]	4	An implementation specific error or failure has occurred.
[XDAS_S_INCOMPLETE_RECORD	5	The audit record has not been fully populated by the caller.
[XDAS_S_INVALID_ACTION_LIST]	6	The action list supplied is not valid.
[XDAS_S_INVALID_AUDIT_STREAM]	7	The audit stream supplied is not valid.
[XDAS_S_INVALID_DAS_REF]	8	The audit service handle supplied does not refer to a valid audit service.
[XDAS_S_INVALID_EVENT_INFO]	9	The specified audit event information is not valid.
[XDAS_S_INVALID_EVENT_NO]	10	The event number supplied is not valid.
[XDAS_S_INVALID_FILTER	11	The filter name supplied is not valid.
[XDAS_S_INVALID_FILTER_EXPR]	12	The filter expression supplied is not valid.
[XDAS_S_INVALID_FILTER_LIST]	13	The list of filter names supplied is not valid.
[XDAS_S_INVALID_FILTER_TYPE]	14	The filter type supplied is not valid.
[XDAS_S_INVALID_INITIATOR_INFO]	15	The initiator information has a syntax error.
[XDAS_S_INVALID_ORIG_INFO]	16	The originator information has a syntax error.
[XDAS_S_INVALID_OUTCOME]	17	The specified outcome is invalid.
[XDAS_S_INVALID_RECORD_DESCRIPTOR]	18	The specified audit record descriptor is not valid.
[XDAS_S_INVALID_RECORD_NUMBER]	19	The specified audit record number is not valid.
[XDAS_S_INVALID_SECURITY_CONTEXT]	20	The security context supplied is invalid.
[XDAS_S_INVALID_TARGET_INFO]	21	The target information has a syntax error.
[XDAS_S_NO_AUDIT]	22	The event does not need to be audited.
[XDAS_S_NO_DECISION_YET]	23	The audit service has insufficient information to decide if the event requires auditing.

Name	Value in Field	Meaning
[XDAS_S_NOT_SUPPORTED]	24	The called function is not supported by this implementation
[XDAS_S_RECORD_SYNTAX_ERROR]	24	A syntax error has been detected in an input record.
[XDAS_S_STORAGE_FAILURE]	25	The audit record cannot be written to stable storage.
[XDAS_S_SERVICE_FAILURE]	26	There has been an audit service failure.

Table 6-2 Routine Errors

All [XDAS_S_*] symbols equate to complete **OM_uint32** status codes, rather than to bit-field values. For example, the actual value of the symbol [XDAS_S_CALL_BAD_STRUCTURE] (value 3 in the **Calling Error** field) is 3 << 16.

The macros:

XDAS_CALLING_ERROR( )
XDAS_ROUTINE_ERROR( )

shall be provided, each of which takes a XDAS status code and removes all but the relevant field. For example, the value obtained by applying XDAS_ROUTINE_ERROR( ) to status code removes the **Calling Errors** field, leaving only the **Routine Errors** field. The values delivered by these macros may be directly compared with a [XDAS_S_*] symbol of the appropriate type. The macro XDAS_ERROR( ) is also provided, which when applied to a XDAS status code returns a non-zero value if the status code indicates a calling or routine error, and a zero value otherwise.

6.8.2 Minor Status Codes

In addition to the function returns, XDAS-API C-language functions return a *minor_status* argument, which is used to indicate specialised errors from the underlying security mechanism. This argument may contain a single mechanism-specific error, indicated by an **OM_uint32** value. It is not expected that portable applications would do anything with this parameter except print out the values for aiding diagnostics.

The *minor_status* argument is always set by a XDAS-API function, even if it returns a calling error or one of the generic API errors indicated above as fatal, although other output arguments may remain unset in such cases. However, output arguments that are expected to return pointers to storage allocated by a function must always be set by the function, even in the event of an error, although in such cases the XDAS-API function may elect to set the returned argument value to NULL to indicate that no storage was actually allocated. Any length field associated with such NULL pointers (as in a **xdas_buffer_desc** structure) shall be set to zero. The XDAS status code [XDAS_S_FAILURE] is used to indicate that the underlying mechanism detected an error for which no specific XDAS status code is defined. The minor status code provides more details about the error.

6.9 Optional Arguments

Various arguments are described as optional. This means that they follow a convention whereby a default value may be requested. The following conventions are used for omitted arguments. These conventions apply only to those arguments that are explicitly documented as optional.

6.9.1 xdas_buffer_t Types *(Input or Input,Output)*

Specify XDAS_C_NO_BUFFER as a value. For an *input* argument this signifies that default behaviour is requested, while for an *input,output* argument it indicates that the information that would be returned by the argument is not required by the application.

6.9.2 Integer Types

Individual argument documentation lists values to be used to indicate default actions. These are passed by value.

6.9.3 Pointer Types

Specify NULL as the value.

6.10 Constants

The tables below set out the manifest constants defined by the specification, and the value to which they are set.

Name	Value	Meaning
XDAS_C_NO_BUFFER	NULL	No buffer is supplied or returned.

Table 6-3 Optional Parameter Constants

6.10.1 Event Record Section Identifiers

These strings are used within an audit event record to mark the beginning and end of sections of data. They are included to support syntax checking.

String Value	Purpose
HDR	Start of header data
ORG	Start of originator data
INT	Start of initiator data
TGT	Start of target data
SRC	Start of pointer to source record
EVT	Start of event specific data
END	End of record

Table 6-4 XDAS Event Record Section Identifiers

6.10.2 Event Numbers

An event number encodes the identification of an event set as well as the identification of the unique event. A set of event numbers is assigned by the OpenGroup (upon request) to an organization or a vendor. The organization or vendor then has the authority to use event numbers within that set.

Conceptually, each event number is a pair (set-id, event-id), where set-id identifies an event set, and the event-id identifies an event within the event set. In practice, each event number must have one of the formats illustrated in Figure 6-2 (on page 50).

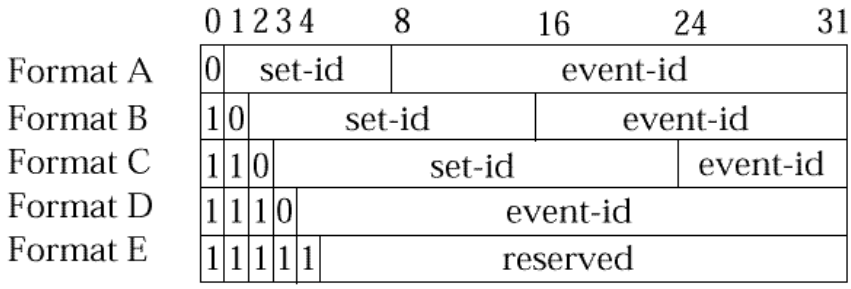


Figure 6-2 Event ID Formats

Given an event number, its format can be determined from its four high-order bits. Format-A event numbers, which are allocated to organizations such as the OpenGroup itself and major vendors which need more than 16 bits for event-number assignment, devote 7 bits to set-id and 24 bits to event-id. Format-B event numbers are allocated to intermediate-size vendors which need 8 to 16 bits for event-number assignment. Format-C event numbers are allocated to small-size vendors which need less than 8 bits for event number assignments. Format-D event numbers are not administered by the OpenGroup and can be used freely for local security domain-specific events. The use of these event numbers might not be unique across cells and should be avoided by servers which may be installed in more than one cells. Format-E event numbers are reserved for future use.

The following table defines the generic set of XDAS events numbers.

XDAS conforming implementations are required to handle all these defined audit events as valid. An application or system developer that submits or imports security domain specific events to the XDAS service must map those events to these XDAS generic events or register their own set of audit events with the OpenGroup.

Request to register audit events should be made by sending email to *dce-registry@osf.org*

Refer to the following URL for a list of registered events and event classes:

http://www.camb.opengroup.org/tech/rfc/rfc81.2.html

An XDAS implementation is required to document the procedures for configuring additional registered audit events into the XDAS service as valid audit events.

Event	Value	Description
XDAS_AE_CREATE_ACCOUNT	0x01000001	Create account
XDAS_AE_DELETE_ACCOUNT	0x01000002	Delete account
XDAS_AE_DISABLE_ACCOUNT	0x01000003	Disable account
XDAS_AE_ENABLE_ACCOUNT	0x01000004	Enable account
XDAS_AE_QUERY_ACCOUNT	0x01000005	Query account attributes
XDAS_AE_MODIFY_ACCOUNT	0x01000006	Modify account attributes
XDAS_AE_CREATE_SESSION	0x01000007	Create a user session
XDAS_AE_TERMINATE_SESSION	0x01000008	Terminate a user session

Event	Value	Description
XDAS_AE_QUERY_SESSION	0x01000009	Query a user session attributes
XDAS_AE_MODIFY_SESSION	0x0100000A	Modify user session attributes
XDAS_AE_CREATE_DATA_ITEM	0x0100000B	Create data item
XDAS_AE_DELETE_DATA_ITEM	0x0100000C	Delete data item
XDAS_AE_QUERY_DATA_ITEM_ATT	0x0100000D	Query data item attributes
XDAS_AE_MODIFY_DATA_ITEM_ATT	0x0100000E	Modify data item attributes
XDAS_AE_INSTALL_SERVICE	0x0100000F	Install service or application
XDAS_AE_REMOVE_SERVICE	0x01000010	Remove service or application
XDAS_AE_QUERY_SERVICE_CONFIG	0x01000011	Query configuration of service or application
XDAS_AE_MODIFY_SERVICE_CONFIG	0x01000012	Modify configuration of service or application
XDAS_AE_DISABLE_SERVICE	0x01000013	Disable service or application
XDAS_AE_ENABLE_SERVICE	0x01000014	Enable service or application
XDAS_AE_INVOKE_SERVICE	0x01000015	Invoke service or application
XDAS_AE_TERMINATE_SERVICE	0x01000016	Terminate service or application
XDAS_AE_QUERY_PROCESS_CONTEXT	0x01000017	Query processing context
XDAS_AE_MODIFY_PROCESS_CONTEXT	0x01000018	Modify processing context
XDAS_AE_CREATE_PEER_ASSOC	0x01000019	Create an association with a peer
XDAS_AE_TERMINATE_PEER_ASSOC	0x0100001A	Terminate an association with a peer
XDAS_AE_QUERY_ASSOC_CONTEXT	0x0100001B	Query an association context
XDAS_AE_MODIFY_ASSOC_CONTEXT	0x0100001C	Modify an association context
XDAS_AE_RECEIVE_DATA_VIA_ASSOC	0x0100001D	Receive data via an association
XDAS_AE_SEND_DATA_VIA_ASSOC	0x0100001E	Send data via an association
XDAS_AE_CREATE_DATA_ITEM_ASSOC	0x0100001F	Create association with data item
XDAS_AE_TERMINATE_DATA_ITEM_ASSOC	0x01000020	Terminate association with data item
XDAS_AE_QUERY_DATA_ITEM_ASSOC_CONTEXT	0x01000021	Query context of association with data item
XDAS_AE_MODIFY_DATA_ITEM_ASSOC_CONTEXT	0x01000022	Modify context of association with data item
XDAS_AE_QUERY_DATA_ITEM_CONTENTS	0x01000023	Query data item contents
XDAS_AE_MODIFY_DATA_ITEM_CONTENTS	0x01000024	Modify data item contents
XDAS_AE_START_SYS	0x01000024	Start system
XDAS_AE_SHUTDOWN_SYS	0x01000025	Shutdown system
XDAS_AE_RESOURCE_EXHAUST	0x01000026	Resource exhaustion
XDAS_AE_RESOURCE_CORRUPT	0x01000027	Resource corruption
XDAS_AE_BACKUP_DATASTORE	0x01000028	Backup datastore
XDAS_AE_RECOVER_DATASTORE	0x01000029	Recover datastore
XDAS_AE_AUD_CONFIG	0x0100002A	Configure audit service
XDAS_AE_AUD_DS_FULL	0x0100002B	Audit datastore full
XDAS_AE_AUD_DS_CORR	0x0100002C	Audit datastore corrupted

Table 6-5 XDAS Event Numbers

6.10.3 XDAS Event Classes

Similar to event numbers, event-class numbers encode the identification of an event-class set as well as the identification of a unique event class within that set. A set of event-class numbers is assigned (upon request) by the OpenGroup to an organization or a vendor. The organization or vendor then has the authority to use the the event-class numbers within that set.

Conceptually, each event class number is a pair (set-id, class-id), where set-id identifies an event-class set, and the class-id identifies an event class within in the set. In practice, each event-class number must have one of the formats illustrated in Figure 6-3.

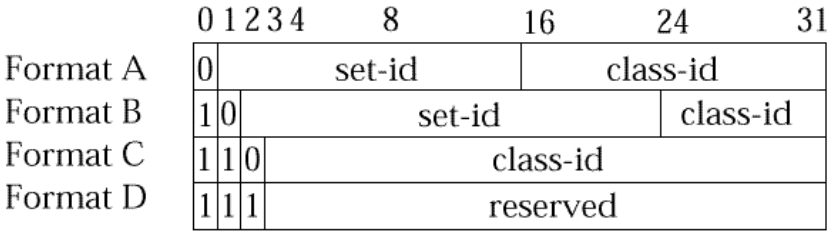


Figure 6-3 Class ID Formats

Given an event-class number, its format can be determined from its three high-order bits. Format-A class numbers, which are allocated to organizations/vendors which need more than 8 bits for event-class number assignment, devote 14 bits to set-id and 16 bits to class-id. Format-B class numbers are allocated to vendors which do not need more than 8 bits for event-class number assignment. Format-C class numbers are not administered by the OpenGroup and can be used freely for local security domain-specific event classes. Format-D numbers are reserved for future use.

Some event classes will be defined by vendors for product-specific events, thus: EC_VendorA_ProductX_EventClass1. Event classes which encompass multiple vendors will be defined by the OpenGroup (e.g., EC_OSF_C2_Configuration), or by security domain administrators who can tailor the definition to satisfy the specific requirements of their security domain.

The default set of event classes are listed in the following table. An XDAS implementation is required to treat as valid all these defined event classes.

The set of event classes may be extended by registration with The OpenGroup, in a similar way to event numbers.

An XDAS implementation is required to document the procedures for configuring additional registered audit event classes into the XDAS service as valid audit event classes.

Event Class	Value	Description
XDAS_AEC_ACCOUNT_MANAGEMENT	0x01000001	Account management events
XDAS_AEC_USER_SESSION	0x01000002	User session events
XDAS_AEC_DATA_ITEM_MANAGEMENT	0x01000003	Data item and resource element management events
XDAS_AEC_SERVICE_MANAGEMENT	0x01000004	Service or application management events
XDAS_AEC_SERVICE_UTILIZE	0x01000005	Service and application utilization events
XDAS_AEC_PEER_ASSOC_MANAGEMENT	0x01000006	Peer association management events
XDAS_AEC_DATA_ITEM_CONTENT_ACCESS	0x01000007	Data item or resource element content access events
XDAS_AEC_EXCEPTIONAL	0x01000008	Exceptional events
XDAS_AEC_AUDIT_SERVICE	0x01000009	Audit service management events

Table 6-6 XDAS Default Event Class Codes

6.11 XDAS Event Outcome Codes

The XDAS outcome codes are:

Name	Value	Meaning
[XDAS_OUT_SUCCESS]	0x00000000	Successful Event
[XDAS_OUT_PRIV_USED]	0x00000100	Privilege used
[XDAS_OUT_PRIV_GRANTED]	0x00000200	Privilege granted
[XDAS_OUT_PRIV_REVOKED]	0x00000400	Privilege revoked
[XDAS_OUT_PRESELECT_CRITERIA_SET]	0x00000800	Preselection criteria set or modified
[XDAS_OUT_THRESHOLDS_SET]	0x00001000	Thresholds set
[XDAS_OUT_ACTIONS_SET]	0x00002000	Actions set for alarms
[XDAS_OUT_THRESHOLD_EXCEEDED]	0x00004000	Pre-set thresholds exceeded
[XDAS_OUT_FAILURE]	0x00000001	Non security relevant failure
[XDAS_OUT_SERVICE_UNAVAILABLE]	0x00000101	Service not available
[XDAS_OUT_SERVICE_FAILURE]	0x00000201	Service failure
[XDAS_OUT_HARDWARE_FAILURE]	0x00000401	Hardware failure or exception condition
[XDAS_OUT_LOST_ASSOCIATION]	0x00001001	Service, user or device already enabled
[XDAS_OUT_ALREADY_DISABLED]	0x00002001	Service, user or device already disabled
[XDAS_OUT_SERVICE_ERROR]	0x00004001	Service returns an error
[XDAS_OUT_BUSY]	0x00008001	Service or device busy
[XDAS_OUT_DISABLED]	0x00010001	Service or device disabled
[XDAS_OUT_INVALID_INPUT]	0x00020001	Input supplied invalid
[XDAS_OUT_ENTITY_EXISTS]	0x00040001	Attempt to create an entity which already exists
[XDAS_OUT_ENTITY_NON-EXISTENT]	0x00080001	Attempt to access a non-existent entity
[XDAS_OUT_DENIAL]	0x00000002	Security relevant failure
[XDAS_OUT_INSUFFICIENT_PRIVILEGE]	0x00000102	Not sufficient privilege
[XDAS_OUT_INVALID_IDENTITY]	0x00000202	Identity supplied not valid
[XDAS_OUT_INVALID_USER_CREDENTIALS]	0x00000402	User credentials supplied are invalid

Table 6-7 XDAS Event Outcome Codes

The outcome codes are structured into sets for SUCCESS, FAILURE, and DENIAL. Multiple codes from within one of these sets may be returned by a single call by combining them using a bitwise OR, but it is not permitted for outcome codes from the different sets to be returned by a single call. That is, multiple SUCCESS codes may returned by one call, but SUCCESS and FAILURE codes may not be returned by a single call.

6.11.1 XDAS Filter Types

The XDAS filter types are:

Name	Value	Meaning
XDAS_C_SUBMIT	1	Filters for event submission interface
XDAS_C_IMPORT	2	Filters for event import interface

Table 6-8 XDAS Filter Types

6.11.2 XDAS Filter Expression Flags

The flags used within filter expressions are:

Name	Value	Meaning
XDAS_C_INCLUDE	1	Include events matching the following rule
XDAS_C_EXCLUDE	2	Exclude events matching the following rule

Table 6-9 XDAS Filter Flags

6.11.3 XDAS Filter Expression Attributes

The attributes that may be used within filter expressions are:

Name	Value	
XDAS_VERSION	1	XDAS Version number
XDAS_TIME_OFFSET	2	Time offset from start of epoch
XDAS_TIME_UNCERT_INTER	3	Time uncertainty interval
XDAS_TIME_UNCERT_INDIC	4	Time uncertainty indicator
XDAS_TIME_SOURCE	5	Source of time
XDAS_TIME_TIME_ZONE	6	Time Zone
XDAS_EVENT_NUMBER	7	Event number
XDAS_OUTCOME	8	Event outcome
XDAS_ORG_LOC_NAME	9	Originator location name
XDAS_ORG_LOC_ADD	10	Originator location address
XDAS_ORG_SERV_TYPE	11	Originator service type
XDAS_ORG_AUTH_AUTH	12	Originator authentication authority
XDAS_ORG_PRINC_NAME	13	Originator principal name
XDAS_ORG_PRINC_IDENTITY	14	Originator principal identity
XDAS_INT_AUTH_AUTH	15	Initiator authentication authority
XDAS_INT_PRINC_NAME	16	Initiator principal name
XDAS_INT_PRINC_IDENTITY	17	Initiator principal identity
XDAS_TGT_LOC_NAME	18	Target location name
XDAS_TGT_LOC_ADD	19	Target location address
XDAS_TGT_SERV_TYPE	20	Target service type
XDAS_TGT_AUTH_AUTH	21	Target authentication authority
XDAS_TGT_PRINC_NAME	22	Target principal name
XDAS_TGT_PRINC_IDENTITY	23	Target principal identity

Table 6-10 XDAS Filter Attributes

6.11.4 XDAS Filter Expression Operators

The operators that may be used within filter expressions are:

Operator	Value	Meaning
XDAS_O_EQ	1	Equal
XDAS_O_NE	2	Not equal
XDAS_O_GT	3	Greater than
XDAS_O_LT	4	Less than
XDAS_O_GE	5	Greater than or equal
XDAS_O_LE	6	Less than or equal
XDAS_O_BT	7	Bitwise test
XDAS_O_SS	8	Substring

Table 6-11 XDAS Filter Operators

6.11.5 XDAS Action Masks

The XDAS action bit masks used within filter definitions are:

Name	Value	Meaning
XDAS_ACT_LOG	1	Record in Audit Stream
XDAS_ACT_ALARM	2	Submit event to Event Management System
XDAS_ACT_ACTION	4	Execute specified command

Table 6-12 XDAS Action Codes

Chapter 7

XDAS Application Program Interface (API)

This chapter presents the functions to be used by callers of the XDAS application programming interfaces

NAME

xdas_close_audit_stream — close the specified audit_stream

SYNOPSIS

```
OM_uint32 xdas_close_audit_stream (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_audit_stream_t   *audit_stream_ref
);
```

DESCRIPTION

xdas_close_audit_stream( ) is a member of the Basic XDAS Conformance class.

The *xdas_close_audit_stream*( ) function closes the audit stream, previously opened for reading, specified by the *audit_stream_ref* handle.

Once an audit stream is closed, using any copy of the handle that refers to that audit stream, that audit stream is no longer valid for use in any XDAS function call.

Note: An implementation may reclaim any storage associated with an audit stream when that audit stream is closed.

The caller must possess the XDAS_AUDIT_READ authority.

If successful, the function returns [XDAS_S_COMPLETE]

The arguments for *xdas_close_audit_stream*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
Handle to the audit service obtained from a previous call to *xdas_initialize_session*( ).

audit_stream_ref (in)
Handle to the audit stream reference which is to be closed.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_AUDIT_STREAM]
The specified audit stream is not valid.

[XDAS_S_INVALID_DAS_REF]
The handle to the audit service is not valid.

ERROR

No other errors are defined.

NAME

xdas_commit_record — write a completed audit record to the audit stream

SYNOPSIS

```
OM_uint32 xdas_commit_record (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_audit_rec_desc_t *audit_record_descriptor

);
```

DESCRIPTION

xdas_commit_record( ) is a member of the Basic XDAS Conformance class.

The *xdas_commit_record*( ) function writes the audit record identified by *audit_record_descriptor* to the current audit stream controlled by the audit service and accessed by *das_ref*. The XDAS implementation adds the time information to the audit record unless a previous call to *xdas_timestamp_record*( ) has been made using *audit_record_descriptor*. The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE].

If any of the event_number, outcome, initiator_information, target_information and event_info parameters to *xdas_start_record*( ) and *xdas_put_event_info*( ) have not been completed in at least one such call, even when component fields are empty, then this call shall return [XDAS_S_INCOMPLETE_RECORD].

The arguments for *xdas_commit_record*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: Handle to the audit service obtained from a previous call to *xdas_initialize_session*( ).

audit_record_descriptor (in)
: A descriptor referring to a completed audit record to be written to the audit stream. On successful completion *audit_record_descriptor* is no longer a valid reference to an audit record.

  Note: An implementation may reclaim any storage associated with an *audit_record_descriptor* when that *audit_record_descriptor* is closed.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INCOMPLETE_RECORD]
: The audit record has not been fully populated by the caller.

[XDAS_S_INVALID_DAS_REF]
The handle to the audit service is not valid.

[XDAS_S_INVALID_RECORD_DESCRIPTOR]
The specified audit record descriptor is not valid.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

[XDAS_S_SERVICE_FAILURE]
There has been an audit service failure.

[DAS_S_STORAGE_FAILURE]
The audit record cannot be written to stable storage.

ERRORS

No other errors are defined.

NAME

xdas_create_filter — create the specified audit filter

SYNOPSIS

```
OM_uint32 xdas_create_filter (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_buffer_t         *name,
    const OM_uint32              filter_type,
    const xdas_buffer_t         *filter_exp,
    const xdas_buffer_t         *filter_act
);
```

DESCRIPTION

xdas_create_filter( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_create_filter*( ) function creates a filter for the *name* specified. If a filter with the specified name already exists the call fails. On creation the filter is in a disabled state.

The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_create_filter*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

name (in)
The name of the filter.

filter_type (in)
The type of filter. This may be either XDAS_C_SUBMIT or XDAS_C_IMPORT.

filter_exp (in)
The expression list which defines the criteria for detection of the event.

filter_act (in) The list of actions to be taken when the event is submitted or imported.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_ACTION_LIST]
The filter type supplied is not recognized.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit daemon.

[XDAS_S_INVALID_FILTER]
The filter name supplied already exists.

[XDAS_S_INVALID_FILTER_EXP]
The filter expression supplied is not valid.

[XDAS_S_INVALID_FILTER_TYPE]
The filter type supplied is not recognized.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_delete_filter — delete the specified audit filter

SYNOPSIS

```
OM_uint32 xdas_delete_filter (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_buffer_t         *name
);
```

DESCRIPTION

xdas_delete_filter( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_delete_filter*( ) function deletes the filter defined by *name* from the XDAS system. This may involve deleting copies of the filter from all agents managed via a particular instance of the XDAS interface. The function does not wait upon the successful deletion of all instances of the filter maintained by XDAS agents. The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_delete_filter*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

name (in)
The name of the filter.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_FILTER]
The filter name supplied is not valid.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_disable_filter — disable the specified filter

SYNOPSIS

```
OM_uint32 das_disable_filter (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_buffer_t         *name
);
```

DESCRIPTION

xdas_disable_filter( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_disable_filter*( ) function disables the filter specified by *name.* It sets the state of the filter to disabled. If necessary the disabled state of the filter may require propagation to all XDAS agents managed by a particular instance of the XDAS Interface. The function does not wait upon the successful disabling of all instances of the filter maintained by XDAS agents. The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_disable_filter*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

name (in)
: The name of the filter to be disabled.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_FILTER]
: The filter name supplied is not known.

[XDAS_S_NOT_SUPPORTED]
: The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_discard_record — discard a previously created audit record

SYNOPSIS

```
OM_uint32 xdas_discard_record (
     OM_uint32                          *minor_status,
     const xdas_audit_ref_t             *das_ref,
     const xdas_audit_desc_t            *audit_record_descriptor
);
```

DESCRIPTION

xdas_discard_record( ) is a member of the Basic XDAS Conformance class.

The *xdas_discard_record*( ) function clears the buffer specified by *audit_record_descriptor* and releases the memory used by it. The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_discard_record*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
Handle to the XDAS service, obtained from a previous call to *xdas_initialize_session*.

audit_record_descriptor (in)
The audit record descriptor returned from a previous call to *xdas_start_record*. On successful completion the *audit_record_descriptor* is no longer a valid reference to an audit record.

Note: An implementation may reclaim any storage associated with an audit record descriptor when that audit record descriptor is discarded.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_RECORD_DESCRIPTOR]
The specified audit record descriptor is not valid.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_enable_filter — enable the specified audit filter

SYNOPSIS

```
OM_uint32 das_enable_filter (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_buffer_t         *name
);
```

DESCRIPTION

xdas_enable_filter( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_enable_filter*( ) function enables the filter corresponding to the *name* specified. If necessary the enabled state of the filter may require propagation to all XDAS agents managed by a particular instance of the XDAS Interface. The function does not wait upon the successful enabling of all instances of the filter maintained by XDAS agents. The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_enable_filter*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

name (in)
: The name of the filter to be enabled.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_FILTER]
: The filter name supplied is not known.

[XDAS_S_NOT_SUPPORTED]
: The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_get_filter — get named audit filter

SYNOPSIS

```
OM_uint32 das_get_filter (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_buffer_t         *name,
    OM_unit32                   *filter_type,
    xdas_buffer_t               *filter_exp,
    xdas_buffer_t               *filter_act,
    OM_uint32                   *filter_status
);
```

DESCRIPTION

xdas_get_filter( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_get_filter*( ) function returns the components of the filter referred to by *name*. The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_get_filter*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

name (in)
: The name of the filter to be returned.

filter_type (out)
: The type of filter. This may be either XDAS_C_SUBMIT or XDAS_C_IMPORT.

filter_exp (out)
: The contents of the expression list that determines the events to be selected by this filter.

filter_act (out)
: The list of actions to be carried out for events selected by this filter.

filter_status (out)
: The enabled or disabled state of the filter. If the filter is enabled then a value of 0 is returned in this parameter, otherwise a value of 1 is returned.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_FILTER]
The filter name supplied is not known.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_get_next — read next set of records from a previously opened audit stream

SYNOPSIS

```
OM_uint32 xdas_get_next (
        OM_uint32                       *minor_status,
        const xdas_audit_ref_t          *das_ref,
        const xdas_audit_stream_t       *audit_stream_ref,
        const OM_unit32                  max_records,
        const xdas_buffer_t             *audit_record_buffer,
        OM_unit32                       *no_of_records
);
```

DESCRIPTION

xdas_get_next( ) is a member of the Basic XDAS Conformance class.

The *xdas_get_next*( ) function copies up to *max-records* complete records from the audit stream accessed by *das_ref* into the buffer *audit_record_buffer* previously allocated by the caller. The actual number of records retrieved by the function is returned in *no_of_records.*

If the function successfully reads a record or records from the audit stream, the cursor associated with the audit stream referred to by *das_ref* will be advanced to the next unread record in the audit stream.

If the call is unsuccessful, the position of the cursor is not changed. The caller must have the XDAS_AUDIT_READ authority

If there are no more available audit records, *no_of_records* is set to 0 and the function returns [XDAS_S_END].

If the size of the buffer *audit_record_buffer* allocated by the caller is too small to hold a single audit record, *no_of_records* is set to 0 and the function returns [XDAS_S_BUFF_TOO_SMALL].

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_get_next*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*( ).

audit_stream_ref (in)
: The handle to the XDAS audit stream, obtained from a previous call to *xdas_open_audit_stream*( ).

max_records (in)"
: The maximum number of records to be returned by the function in any one call.

audit_record_buffer (in)
: Pointer to the buffer to which the audit records are to be copied.

no_of_records (out)
: the number of records actually copied into *audit_record_buffer.*

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_BUFF_TOO_SMALL]
The buffer allocated by the caller is too small to hold a single audit record.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_END]
The end of the audit stream has been reached.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_STREAM_REF]
The audit stream handle supplied is invalid.

ERRORS

No other errors are defined.

NAME

xdas_import_event_records — imports records from an external audit service into XDAS in XDAS common format

SYNOPSIS

```
OM_uint32 xdas_import_event_records (
        OM_uint32                       *minor_status,
        const xdas_audit_ref_t          *das_ref,
        const xdas_buffer_t             *audit_record_buffer,
        OM_uint32                       *position_in_buffer
);
```

DESCRIPTION

xdas_import_event_records( ) is a member of the XDAS Import API Option Conformance class.

The *xdas_import_event_records*( ) function allows a caller to import audit event records in the XDAS format directly to the XDAS service. The caller places one or more complete audit event records into the buffer referred to by *audit-record_buffer* from which they are copied and integrated into the XDAS audit stream. The function reads audit records until the start of a next record is not found. The implementation may select the records that are actually imported based upon some selection criteria. The caller is not advised of the disposition of the audit records it submits.

The caller must possess the XDAS_AUDIT_IMPORT authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_import_event_records*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: Handle to the XDAS service obtained by a previous call to *xdas_initialize_session*( ).

audit_record_buffer (in)
: Buffer into which the caller places the audit records to be imported into the XDAS audit stream.

position_in_buffer (out)
: If a record syntax error is detected this parameter contains the position in the buffer at which the syntax error was detected.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

[XDAS_S_RECORD_SYNTAX_ERROR]
A syntax error has been detected in an input record.

ERRORS

No other errors are defined.

NAME

xdas_initialize_session — initialize a session with the distributed audit service

SYNOPSIS

```
OM_uint32 xdas_initialize_session (
        OM_uint32                       *minor_status,
        const xdas_buffer_t             *security_context,
        const xdas_buffer_t             *org_info,
        xdas_audit_ref_t                *das_ref
);
```

DESCRIPTION

xdas_initialize_session( ) is a member of the Basic XDAS Conformance class.

The *xdas_initialize_session*( ) function initiates a session between the caller identified by *org_info* and the distributed audit service. The *org_info* is inserted by the implementation into every audit record submitted by the caller via subsequent calls to XDAS functions within the XDAS session. *xdas_initialize_session*( ) validates the *security_context* provided to ensure that caller has been authenticated and is authorized to use the XDAS.

If successful, the function returns *das_ref*, a handle to the XDAS server. The caller must have the XDAS_AUDIT_SERVICE authority.

If successful, the function returns [XDAS_S_COMPLETE].

The use of this function must itself be audited by the XDAS service.

The arguments for *xdas_initialize_session*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

security_context (optional,in)
: An optional opaque structure defining the security context of the caller requesting use of the audit service. This is used to authenticate the caller to the XDAS and establish the callers XDAS authorizations. If this parameter is set to XDAS_NO_BUF, an XDAS implementation shall retrieve default security context from the process environment. The representation of a non-XDAS_NO_BUF security context is implementation defined.

org_info (in)
: This buffer includes the originator information that is to be included with each audit event subsequently submitted by this caller. The XDAS service uses this information to populate the originator information of an audit record when *xdas_start_record*( ) is invoked.

das_ref (out)
: The handle to the XDAS server is returned in *das_ref*.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_ORIG_INFO]
The originator information supplied has a syntax error.

[XDAS_S_INVALID_SECURITY_CONTEXT]
The security context supplied is not valid.

ERRORS

No other errors are defined.

NAME

xdas_list_filters — list the audit filters that have been defined

SYNOPSIS

```
OM_uint32 xdas_list_filters (
    OM_uint32                       *minor_status,
    const xdas_audit_ref_t          *das_ref,
    xdas_buffer_t                   **filter_name_list
);
```

DESCRIPTION

xdas_list_filters( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_list_filters*( ) function yields a pointer to an array of pointers to buffers holding the names of the filters. The last pointer in the array is NULL. The caller must possess the XDAS_AUDIT_CONTROL authority.

The memory for holding the array of pointers and the name filter buffers is allocated by xdas_list_filters( ). When the list of filter names is no longer required, the caller should call *xdas_release_filter_list*( ) to release the memory used to hold the list of filter names.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_list_filters*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

filter_name_list (out)
The list of the names of the filters that exist within the XDAS service.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The handle to the XDAS server supplied does not point to the audit service.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined

NAME

xdas_open_audit_stream — open the audit_stream

SYNOPSIS

```
OM_uint32 xdas_open_audit_stream (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    xdas_audit_stream_t         **audit_stream_ref
);
```

DESCRIPTION

xdas_open_audit_stream( ) is a member of the Basic XDAS Conformance class.

The *xdas_open_audit_stream*( ) function opens the audit stream for reading and returns a handle to the audit stream in *audit_stream_ref* handle. A caller may obtain more than one handle to the audit stream, each of which is independent of any other handles. The caller must possess the XDAS_AUDIT_READ authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_open_audit_stream*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: Handle to the audit service obtained from a previous call to *xdas_initialize_session*.

audit_stream_ref (out)
: Handle to the audit stream returned by the function.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The handle to the audit service is not valid.

ERROR

No other errors are defined.

NAME

xdas_parse_record — parse an audit event record in an audit record buffer

SYNOPSIS

```
OM_uint32 xdas_parse_record (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_audit_desc_t     *audit_record_buffer,
    const OM_uint32              record_number,
    xdas_audit_record_t         *audit_record
);
```

DESCRIPTION

xdas_parse_record( ) is a member of the XDAS Conformance class.

The *xdas_parse_record*( ) function parses and decomposes record number *record_number* in *audit_record_buffer* filled with a number of records by a previous call to *xdas_get_next*( ). Records are extracted from *audit_record_buffer* by starting with record number 0 and interating until one less than the number of records returned by *xdas_get_next*( ).

If successful, the function returns [XDAS_S_COMPLETE].

If *record_number* does not match a record within *audit_record_buffer* then [XDAS_S_INVALID_RECORD_NUMBER] is returned.

The arguments for *xdas_parse_record*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*( ).

audit_record_buffer (in)
The handle to the audit record buffer populated by a previous call to *xdas_get_next*( ).

record_number (in)
The number of the record to be retrieved from *audit_record_buffer*. The first record is referred to by a *record_number* of 0. *audit_record* (in,out)"
The structure containing the fields of the parsed audit record.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_EVENT_NO]
The specified event number is not valid.

[XDAS_S_INVALID_INITIATOR_INFO]
The initiator information supplied has a syntax error.

[XDAS_S_INVALID_OUTCOME]
The specified outcome is not valid.

[XDAS_S_INVALID_RECORD_NUMBER]
The specified audit record number is not valid.

[XDAS_S_INVALID_TARGET_INFO]
The specified target information has a syntax error.

[XDAS_S_NO_AUDIT]
The event specified does not need to be audited.

[XDAS_S_NO_DECISION_YET]
The audit service has insufficient information to decide if the event requires auditing.

[XDAS_S_INVALID_EVENT_INFO]
The specified audit event information is not valid.

ERRORS

No other errors are defined.

NAME

xdas_put_event_info — add specific event information to an audit record buffer

SYNOPSIS

```
OM_uint32 xdas_put_event_info (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    const xdas_audit_desc_t     *audit_record_descriptor,
    const OM_unit32              event_number,
    const OM_unit32              outcome,
    const xdas_buffer_t         *initiator_information,
    const xdas_buffer_t         *target_information,
    const xdas_buffer_t         *event_information
);
```

DESCRIPTION

xdas_put_event_info( ) is a member of the XDAS Event Submission API Option Conformance class.

The *xdas_put_event_info*( ) function adds event information to an audit record or overwrites existing information. If the combination of information submitted and already present in the audit record referred to by *audit_record_descriptor*, is insufficient to evaluate applicable pre-selection criteria, the function returns XDAS_S_NO_DECISION_YET to the caller. If there is sufficient information for evaluation of applicable pre-selection checks the XDAS_S_COMPLETE or XDAS_S_NO_AUDIT are returned to the caller. Multiple calls to *xdas_put_event_info* may be made. For any individual parameter, information supplied in this call will overwrite any previous information supplied.

Although several parameters are optional in this call, a caller shall have populated all the parameters, even when empty, in one or more sequences of calls to *xdas_start_record*( ) and *xdas_put_event-info*( ) before a call to *xdas_commit_record*( ) shall be successful.

The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE], [XDAS_S_NO_DECISION_YET] or [XDAS_S_NO_AUDIT]. If [XDAS_S_NO_AUDIT] is returned, *audit_record_descriptor*. is no longer a valid reference to an audit record.

Note: An implementation may reclaim any storage associated with the audit record descriptor when [XDAS_S_NO_AUDIT] is returned.

If [XDAS_S_NO_DECISION_YET] is returned, then the caller should continue to construct the audit record by subsequent calls to *xdas_put_event_info*( ).

The arguments for *xdas_put_event_info*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*( ).

audit_record_descriptor (in)
: The handle to the audit record, obtained from a previous call to *xdas_start_record*( ).

event_number (optional,in)
: The event number of the detected event. If specified as a NULL value then the event number currently associated with *audit_record_descriptor* is unchanged. Otherwise

event_number overwrites the current value. Only event numbers configured as registered by the implementation shall be valid. Any other event number shall result in the return of [XDAS_S_INVALID_EVENT_NO].

outcome (optional,in)
The outcome of the event determined by the caller. If specified as a NULL value then the outcome code currently associated with *audit_record_descriptor* is unchanged by this call. Otherwise *outcome_code* overwrites the current value. Only the outcome codes listed in Table 6-7 (on page 54) are valid.

initiator_information (optional,in)
The information describing the initiator in the format required by the XDAS common audit format. If specified as XDAS_C_NO_BUFFER the current initiator information associated with the *audit_record_descriptor* supplied is unchanged by this call. Otherwise the contents of *initiator_information* overwrite the current value associated with *audit_record_descriptor*.

target_information (optional,in)
The information on the target of the event in the format required by the XDAS common audit format. If specified as XDAS_C_NO_BUFFER the current target information associated with the *audit_record_descriptor* supplied is unchanged by this call. Otherwise the contents of *target_information* overwrite the current value associated with *audit_record_descriptor*.

event_information (optional,in)
The event specific information that is to be added to the audit record specified by *audit_record_descriptor*. If specified as XDAS_C_NO_BUFFER the current event specific information associated with the *audit_record_descriptor* supplied is unchanged by this call. Otherwise the contents of *event_information* overwrite the current value associated with *audit_record_descriptor*.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_EVENT_INFO]
The specified audit event information is not valid.

[XDAS_S_INVALID_EVENT_NO]
The specified event number is not valid.

[XDAS_S_INVALID_INITIATOR_INFO]
The initiator information supplied has a syntax error.

[XDAS_S_INVALID_OUTCOME]
The specified outcome is not valid.

[XDAS_S_INVALID_RECORD_DESCRIPTOR]
The specified audit record descriptor is not valid.

[XDAS_S_INVALID_TARGET_INFO]
The specified target information has a syntax error.

[XDAS_S_NO_AUDIT]
The event specified does not need to be audited.

[XDAS_S_NO_DECISION_YET]
The audit service has insufficient information to decide if the event requires auditing.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_release_buffer — free storage associated with a buffer

SYNOPSIS

```
OM_uint32 xdas_release_buffer(
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    xdas_buffer_t               *buffer
);
```

DESCRIPTION

xdas_release_buffer( ) is a member of the Basic XDAS Conformance class.

The *xdas_release_buffer*( ) function frees storage associated with a buffer. The storage must have been allocated by an XDAS-API function. In addition to freeing the associated storage, the function zeros the length field in the *buffer* argument. If successful, the function returns [XDAS_S_COMPLETE]. The arguments for *xdas_release_buffer*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: Handle to the XDAS service obtained by a previous call to *xdas_initialize_session*( ).

buffer (in,out)
: The storage associated with the *buffer* is deleted. The xdas_buffer_t object is not freed, but its length field is zeroed.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied is invalid.

ERRORS

No other errors are defined.

NAME

xdas_release_filter_list — release the list of filter names

SYNOPSIS

```
OM_uint32 das_release_filter_list (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    xdas_buffer_t               **filter_list
);
```

DESCRIPTION

xdas_release_filter_list( ) is a member of the XDAS Filter Management API Option Conformance class.

The *xdas_release_filter_list*( ) function releases the list of filter names which were obtained by a previous call to *xdas_list_filters*.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_release_filter_list*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

filter_list (in)
: The pointer to the array of filter names obtained from a previous call to *xdas_list_filters*( ).

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_FILTER_LIST]
: The list of filter names is not valid.

[XDAS_S_NOT_SUPPORTED]
: The called function is not supported by this implementation.

ERRORS

No other errors are defined.

NAME

xdas_rewind_audit_stream — rewind the audit stream

SYNOPSIS

```
OM_uint32 xdas_rewind_audit_stream (
    OM_uint32                   *minor_status,
    const xdas_audit_ref_t      *das_ref,
    xdas_audit_stream_t         *audit_stream_ref
);
```

DESCRIPTION

xdas_rewind_audit_stream( ) is a member of the Basic XDAS Conformance class.

The *xdas_rewind_audit_stream*( ) function rewinds the audit stream referred to by *xdas_stream_ref* so that the cursor associated with the *xdas_stream_ref* points to the first record in the audit stream. The caller must possess the XDAS_AUDIT_READ authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_rewind_audit_stream*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*( ).

audit_stream_ref (in,out)
: Handle to the audit stream which is to be rewound.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_AUDIT_STREAM]
: The specified audit stream is not valid.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not point to the audit service.

ERRORS

No other errors are defined.

NAME

xdas_start_record — initialize an audit record

SYNOPSIS

```
OM_uint32 xdas_start_record (
        OM_uint32                   *minor_status,
        const xdas_audit_ref_t      *das_ref,
        xdas_audit_desc_t           *audit_record_descriptor,
        const OM_unit32              event_number,
        const OM_uint32              outcome,
        const xdas_buffer_t         *initiator_information,
        const xdas_buffer_t         *target_information,
        const xdas_buffer_t         *event_info
);
```

DESCRIPTION

xdas_start_record( ) is a member of the XDAS Event Submission API Option Conformance class.

The *xdas_start_record*( ) function returns a *audit_record_descriptor* handle to the audit record to the caller. If the optional parameters are not specified in the call, then the audit record is initialized but requires fully populating by subsequent calls to *xdas_put_event_info*.

If the optional parameters are specified, *xdas_start_record* determines whether a specified event should be audited given the *event_number*, *outcome* and *initiator_information* supplied. If the event should be audited a valid *audit_record_descriptor* is returned to the caller. If the audit event does not require auditing then *audit-record_descriptor* is set to NULL. The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE].

Although several parameters are optional in this call, a caller shall have populated all the parameters, even when empty, in one or more sequences of calls to *xdas_start_record*( ) and *xdas_put_event-info*( ) before a call to *xdas_commit_record*( ) shall be successful.

The arguments for *xdas_start_record*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: Handle to the XDAS service, obtained from a previous call to *xdas_initialize_session*( ).

audit_record_descriptor (out)
: Pointer to an audit record, populated as defined by the optional input parameters. If the event does not need to be audited, a NULL pointer is returned.

event_number (optional,in)
: The event_number of the detected event.

outcome (optional,in)
: The outcome of the event as determined by the caller.

initiator_information (optional,in)
: The available information describing the initiator in the format required by the XDAS common audit format.

target_information (optional,in)
: Information on the target of the event in the format required by the XDAS common audit

format.

event_info (optional,in)
Information specific to the event.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not point to the audit service.

[XDAS_S_INVALID_EVENT_NO]
The event number specified is not valid.

[XDAS_S_INVALID_INITIATOR_INFO]
The initiator information specified has a syntax error.

[XDAS_S_INVALID_OUTCOME]
The outcome supplied is not valid.

[XDAS_S_INVALID_TARGET_INFO]
The target information specified has a syntax error.

[XDAS_S_INVALID_EVENT_INFO]
The event information specified is not valid.

[XDAS_S_NO_AUDIT]
The specified event does not need to be audited.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

[XDAS_S_UNCERTAIN_AUDIT]
There is uncertainty as to whether the specified event requires auditing.

ERRORS

No other errors are defined.

NAME

xdas_terminate_session — terminate a session with the distributed audit service

SYNOPSIS

```
OM_uint32 das_terminate_session (
    OM_uint32                    *minor_status,
    const xdas_audit_ref_t       *das_ref
);
```

DESCRIPTION

xdas_terminate_session( ) is a member of the Basic XDAS Conformance class.

The *xdas_terminate_session*( ) closes a session between the caller and the distributed audit service. The caller must have the XDAS_AUDIT_SERVICE authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_terminate_session*( ) are:

minor_status (out)
: An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
: The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
: The caller does not possess the required authority.

[XDAS_S_COMPLETE]
: Successful completion.

[XDAS_S_FAILURE]
: An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
: The audit service handle supplied does not represent a valid audit service session.

ERRORS

No other errors are defined.

NAME

xdas_timestamp_record — timestamp the supplied audit record

SYNOPSIS

```
OM_uint32 das_timestamp_record (
    OM_uint32                    *minor_status,
    const xdas_audit_ref_t       *das_ref,
    const xdas_audit_desc_t      *audit_record_descriptor
);
```

DESCRIPTION

xdas_timestamp_record( ) is a member of the XDAS Event Submission API Option Conformance class.

The *xdas_timestamp_record*( ) puts a timestamp on the audit record supplied. The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_timestamp_record*( ) are:

minor_status (out)
An implementation specific return status that provides additional information when [XDAS_S_FAILURE] is returned by the function.

das_ref (in)
The handle to the XDAS server, obtained from a previous call to *xdas_initialize_session*.

audit_record_descriptor (in)
The handle to the audit record returned from a previous call to *xdas_start_record*( ).

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_AUTHORIZATION_FAILURE]
The caller does not possess the required authority.

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_FAILURE]
An implementation specific error or failure has occurred.

[XDAS_S_INVALID_DAS_REF]
The audit service handle supplied does not represent a valid audit service session.

[XDAS_S_INVALID_RECORD_DESCRIPTOR]
The specified audit record descriptor is not valid.

[XDAS_S_NOT_SUPPORTED]
The called function is not supported by this implementation.

ERRORS

No other errors are defined.

Appendix A

Example XDAS Event Mappings

A.1 Oracle Security Events

The following events have been taken from the Oracle Database Administrator's Manual. The table below presents an illustrative mapping to XDAS events.

Table A-1 Mapping of ORACLE Audit Events to XDAS Generic Audit Events

Oracle Event Description	XDAS-API Event(s)
Alter system	configure service or application
Create / drop cluster	configure service or application
Alter / truncate cluster	configure service or application
Create / drop database link	configure service or application
Create / delete index	create / delete data item
Alter index	modify data item
Not exists	THIS IS REPRESENTED BY AN OUTCOME CODE
Create / replace function	configure service or application
Create / replace package / package body	configure service or application
Create / replace procedure	configure service or application
Drop function, package, procedure	configure service or application
Create / drop public database link	configure service or application
Create / drop public synonym	configure service or application
Create / drop role	configure service or application
Set / alter role	configure service service or application
Create / drop rollback segment	create / delete data item
Alter rollback segment	configure service
Create / drop sequence	create / delete data item
Session connect / disconnect	create / terminate an association
Set system audit	configure audit service
System grant	modify account attributes
Create / drop table	create / delete data item
Truncate table	modify data item contents
Create / drop tablespace	configure service or application
Alter tablespace	configure service or application
Create trigger	configure service or application
Alter trigger enable / disable	modify data item
Create / drop / alter user	create / delete / modify account
Create / drop view	create / delete data item
Alter sequence	modify data item
Alter table, comment on table	modify data item
Execute procedure	invoke service or application
Grant / revoke privilege on procedure	configure service or application

Oracle Event Description	**XDAS-API Event(s)**
Grant/revoke privilege on sequence	configure service or application
Grant/revoke privilege on table	modify data item attributes
Insert into table	modify data item
Lock table	modify data item attributes
Select sequence, table	create association with data item
Update table, view	modify data item
Upgrade data	modify data item attributes
Downgrade data	modify data item attributes
Upgrade higher level rows	modify data item attributes
Insert, update, delete lower level rows	create/delete data items, modify data item attributes
Lower DBMS label	modify data item attributes
Raise DBMS label	modify data item attributes
Alter DBMS label to a non-comparable label	modify data item attributes
Grant MAC privileges	modify account attributes, modify an association context
Switch modes	modify an association context

A.2 Mapping

The following events have been taken from the SUN Solaris BSM Manual for audit records. The table below shows where they map to the suggested XDAS events.

Table A-2 Mapping of Solaris BSM Audit Events to XDAS Generic Audit Events

BSM Kernel-level Audit Events	XDAS-API Event
access(2)	query data item attributes
acct(2)	configure audit service
adjtime(2)	configure service or application
chdir(2)	modify processing context
chmod(2)	modify data item attributes
chown(2)	modify data item attributes
chroot(2)	modify processing context
close(2)	terminate association with data item
creat(2)	create data item
exec(2)	invoke service or application component
execve(2)	as exec(2)
exit(2)	terminate service or application component
fchdir(2)	modify processing context
fchmod(2)	modify data item attributes
fchown(2)	modify data item attributes
fchroot(2)	modify processing context
fcntl(2)	modify data item attributes
fork(2)	invoke service or application
fstat(2)	query data item attributes
fstatfs(2)	query configuration of service or application
ioctl(2)	modify data item attributes
kill(2)	modify data item contents
link(2)	modify data item attributes
lstat(2)	query data item attributes
mkdir(2)	create data item
mknod(2)	create data item
mmap(2)	create a data item
mount(2)	enable service
msgctl(2)	modify data item attributes
msgget(2)	create data item,
msgrcv(2)	query data item contents
msgsnd(2)	modify data item contents
munmap(2)	delete data item
open(2)	create an association with a data item
pathconf(2)	query context of association with data item
pipe(2)	create a data item
process dumped core	resource corruption
readlink(2)	query data item contents
rename(2)	modify data item,
rmdir(2)	delete data item
semctl(2)	modify data item attributes

BSM Kernel-level Audit Events	XDAS-API Event
semget(2)	create data item
semop(2)	query/modify data item contents
setgroups(2)	modify user session attributes
setspgrp(2)	modify user session attributes
setrlimit(2)	query/modify configuration of service or application
shmat(2)	create association with peer
shmctl(2)	query/modify data item attributes
shmdt(2)	terminate association with peer
shmget(2)	create data item
stat(2)	query data item attributes
statfs(2)	query configuration of service or application
symlink(2)	modify data item attributes
system(2)	invoke a service or application
umount(2)	terminate a service or application
unlink	modify data item attributes
utimes	modify data item attributes
vfork(2)	invoke service or application
vtrace(2)	invoke service or application
/usr/sbin/allocate	enable or disable devices
/usr/sbin/halt	shutdown system
/usr/sbin/inetd	create an association with a peer
/usr/sbin/in.ftpd	creat an association with a peer
/usr/bin/login	create user session
/usr/lib/nfs/mountd	modify configuration of service or application
/usr/bin/passwd	modify account attributes
/usr/sbin/reboot	start system
/usr/sbin/in.rshd	or create user session
/usr/bin/su	modify user session attributes

A.3 IEEE P1003.1e -- Protection, Audit and Control Interfaces

This table maps the audit events defined in IEEE P1003.1e Draft 15 with the generic XDAS events.

P1003.1e Audit Event	XDAS-API Event
AUD_AET_AUD_SWITCH	Configure audit service
AUD_AET_AUD_WRITE	access to other services
AUD_AET_CHDIR	modify processing context
AUD_AET_CHMOD	modify data item attributes
AUD_AET_CHOWN	modify data item attributes
AUD_AET_CREAT	create a data item
AUD_AET_DUP	create association with a data item
AUD_AET_EXEC	invoke service or application
AUD_AET_EXIT	terminate service or application
AUD_AET_FORK	invoke service or application
AUD_AET_KILL	terminate service or application
AUD_AET_LINK	modify data item attributes
AUD_AET_MKDIR	create data item
AUD_AET_MKFIFO	create data item
AUD_AET_OPEN	create association with data item
AUD_AET_PIPE	create data item
AUD_AET_RENAME	modify data item contents
AUD_AET_RMDIR	delete data item
AUD_AET_SETGID	modify user session attributes
AUD_AET_SETUID	modify user session attributes
AUD_AET_UNLINK	modify data item attributes
AUD_AET_UTIME	modify data item attributes
AUD_AET_ACL_DELETE_DEF_FILE	modify data item attributes
AUD_AET_ACL_SET_FD	modify data item attributes
AUD_AET_ACL_SET_FILE	modify data item attributes
AUD_AET_CAP_SET_FD	modify data item attributes
AUD_AET_CAP_SET_FILE	modify data item attributes
AUD_AET_CAP_SET_PROC	modify processing context
AUD_AET_INF_SET_FD	modify data item attributes
AUD_AET_INF_SET_FILE	modify data item attributes
AUD_AET_INF_SET_PROC	modify processing context
AUD_AET_MAC_SET_FD	modify data item attributes
AUD_AET_MAC_SET_FILE	modify data item attributes
AUD_AET_MAC_SET_PROC	modify processing context

Table A-3 Mapping of IEEE P1003.1e Audit Events to XDAS Generic Audit Events

Glossary

API

Application Programming Interface.

The interface between the application software and the application platform, across which all services are provided.

The application programming interface is primarily in support of application portability, but system and application interoperability are also supported by a communication API (see POSIX.0).

audit

See Security Audit (see ISO / IEC 7498-2).

audit analysis

The *analysis* of audit data comprises manual or automated processes which scrutinize the audit data to identify in them real or potential security threats or to track system activity for the purpose of assigning accountability. Several approaches are possible including:

- to compare activity with a profile based on *normal* behaviour;
- to seek out unacceptable or suspicious events by establishing a rules base for inappropriate system activity.

Analysis can generate filtering requirements which can be fed back into the discrimination process and provide strong reporting utilities.

audit trail

See Security Audit Trail (see ISO / IEC 7498-2).

authenticated identity

An identity of a principal that has been assured through authentication (see ISO / IEC 10081-2).

authentication

Verify claimed identity; see data origin authentication, and peer entity authentication (see ISO / IEC 7498-2). Authentication information in the form of a security certificate which may be used to assure the identity of an entity guaranteed by an authentication authority (see ISO / IEC 10081-2).

authorization

The granting of rights, which includes the granting of access based on access rights (see ISO / IEC 7498-2).

authorization policy

A set of rules, part of an access control policy, by which access by security subjects to security objects is granted or denied. An authorization policy may be defined in terms of access control lists, capabilities or attributes assigned to security subjects, security objects or both (see ECMA TR / 46).

availability

The property of being accessible and usable upon demand by an authorized entity (see ISO / IEC 7498-2).

client-server
These operations occur between a pair of communicating independent peer processes. The peer process initiating a service request is termed the client. The peer process responding to a service request is termed the server. A process may act as both client and server in the context of a set of transactions.

confidentiality
The property that information is not made available or disclosed to unauthorized individuals, entities, or processes (see ISO / IEC 7498-2).

credentials
Data that is transferred to establish the claimed identity of an entity (see ISO / IEC 7498-2).

data integrity
The property that data has not been altered or destroyed in an unauthorized manner (see ISO / IEC 7498-2).

denial of service
The unauthorized prevention of authorized access to resources or the delaying of time-critical operations (see ISO / IEC 7498-2).

identification
The assignment of a name by which an entity can be referenced. The entity may be high level (such as a user) or low level (such as a process or communication channel.

initiator
An entity (for example, human user or computer based entity) that attempts to access other entities (see ISO / IEC 10081-3).

integrity
See Data Integrity (see ISO / IEC 7498-2).

policy
See security policy (see ISO / IEC 7498-2).

privacy
The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Note: because this term relates to the right of individuals, it cannot be very precise and its use should be avoided except as a motivation for requiring security (see ISO / IEC 7498-2).

security attribute
A security attribute is a piece of security information which is associated with an entity.

security audit
An independent review and examination of system records and operations in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security and to recommend any indicated changes in control, policy and procedures (see ISO / IEC 7498-2).

security audit message
A message generated following the occurrence of an auditable security-related event (see ISO / IEC 10081-7).

security audit record
A single record in a security audit trail corresponding to a single security-related event (see ISO / IEC 10081-7).

security audit trail
Data collected and potentially used to facilitate a security audit (see ISO / IEC 7498-2).

security auditor
An individual or a process allowed to have access to the security audit trail and to build audit reports (see ISO / IEC 10081-7).

security domain
A set of elements, a security policy, a security authority and a set of security-relevant operations in which the set of elements are subject to the security policy, administered by the security authority, for the specified operations (see ISO / IEC 10081-1).

security policy
The set of criteria for the provision of security services (see also identity-based and rule-based security policy).

security service
A service which may be invoked directly or indirectly by functions within a system that ensures adequate security of the system or of data transfers between components of the system or with other systems.

target
An entity to which access may be attempted (see ISO / IEC 10081-3).

threat
A potential violation of security (see ISO / IEC 7498-2).
An action or event that might prejudice security (see ITSEC).

trust
A relationship between two elements, a set of operations and a security policy in which element X trusts element Y if and only if X has confidence that Y behaves in a well defined way (with respect to the operations) that does not violate the given security policy (see ISO / IEC 10081-1).

trusted functionality
That which is perceived to be correct with respect to some criteria, for example, as established by a security policy (see ISO / IEC 7498-2).

Index