Preliminary Specification

Distributed Audit Service (XDAS)

Company Review Version

The Open Group

Preliminary Specification

Distributed Audit Service (XDAS) Company Review Version

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

Published in the U.K. by The Open Group, February 1997.

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 is an international open systems organization that is leading the way in creating the infrastructure needed for the development of network-centric computing and the information superhighway. Formed in 1996 by the merger of the X/Open Company and the Open Software Foundation, The Open Group is supported by most of the world's largest user organizations, information systems vendors and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to assist user organizations, vendors and suppliers in the development and implementation of products supporting the adoption and proliferation of open systems.

With more than 300 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- consolidating, prioritizing and communicating customer requirements to vendors
- conducting research and development with industry, academia and government agencies to deliver innovation and economy through projects associated with its Research Institute
- managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- adopting, integrating and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- licensing and promoting the X/Open brand that designates vendor products which conform to X/Open Product Standards
- promoting the benefits of open systems to customers, vendors and the public.

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trade mark on behalf of the industry.

The X/Open Process

This description is used to cover the whole Process developed and evolved by X/Open. It includes the identification of requirements for open systems, development of CAE and Preliminary Specifications through an industry consensus review and adoption procedure (in parallel with formal standards work), and the development of tests and conformance criteria.

This leads to the preparation of a Product Standard which is the name used for the documentation that records the conformance requirements (and other information) to which a vendor may register a product. There are currently two forms of Product Standard, namely the Profile Definition and the Component Definition, although these will eventually be merged into one.

The X/Open brand logo is used by vendors to demonstrate that their products conform to the relevant Product Standard. By use of the X/Open brand they guarantee, through the X/Open Trade Mark Licence Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the vendor.

Open Group Publications

The Open Group publishes a wide range of technical literature, the main part of which is focused on specification development and product documentation, but which also includes Guides, Snapshots, Technical Studies, Branding and Testing documentation, industry surveys and business titles.

There are several types of specification:

- *CAE Specifications*

  CAE (Common Applications Environment) Specifications are the stable specifications that form the basis for our product standards, which are used to develop X/Open branded systems. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement a CAE Specification can enjoy the benefits of a single, widely supported industry standard. In addition, they can demonstrate product compliance through the X/Open brand. CAE Specifications are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

- *Preliminary Specifications*

  Preliminary Specifications usually address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary Specification is not a draft specification; rather, it is as stable as can be achieved, through applying The Open Group's rigorous development and review procedures.

  Preliminary Specifications are analogous to the *trial-use* standards issued by formal standards organizations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a CAE Specification. While the intent is to progress Preliminary Specifications to corresponding CAE Specifications, the ability to do so depends on consensus among Open Group members.

- *Consortium and Technology Specifications*

  The Open Group publishes specifications on behalf of industry consortia. For example, it publishes the NMF SPIRIT procurement specifications on behalf of the Network Management Forum. It also publishes Technology Specifications relating to OSF/1, DCE, OSF/Motif and CDE.

  Technology Specifications (formerly AES Specifications) are often candidates for consensus review, and may be adopted as CAE Specifications, in which case the relevant Technology Specification is superseded by a CAE Specification.

In addition, The Open Group publishes:

- *Product Documentation*

  This includes product documentation — programmer's guides, user manuals, and so on — relating to the Pre-structured Technology Projects (PSTs), such as DCE and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

- *Guides*

  These provide information that is useful in the evaluation, procurement, development or management of open systems, particularly those that relate to the CAE Specifications. The Open Group Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming conformance to a Product Standard.

- *Technical Studies*

  Technical Studies present results of analyses performed on subjects of interest in areas relevant to The Open Group's Technical Program. They are intended to communicate the findings to the outside world so as to stimulate discussion and activity in other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of a technical activity.

Versions and Issues of Specifications

As with all *live* documents, CAE 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.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at **http://www.opengroup.org/public/pubs**.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at **http://www.opengroup.org/public/pubs**.

This Document

This document is a Preliminary Specification (see above).

- Chapter 1 is an introduction to the GAS-API.
- 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 DAS 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,
- Appendix B describes the syntax used for names 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

OSF^{TM} is a trademark of The Open Software Foundation, Inc.

X/Open® is a registered trademark, and the ‘‘X’’ device is a trademark, of X/Open Company Limited.

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:

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 X / Open documents are referenced in this specification:

XDSF
: Guide, December 1994, Distributed Security Framework (ISBN: 1-85912-071-7, G410).

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

XEMS
: X / Open 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 unauthorised individuals to access the system and of attempts by authorised 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 utilise 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 between components.
- 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 pre-selection 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 pre-selection 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 analyse 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 organisations benefit through increased effectiveness in enforcing individual accountability within a distributed environment.

1.1 Functional Requirements

The business requirements for a distributed audit service are detailed in this section for completeness. Not all of these requirements are satisfied by the current scope of XDAS. The requirements are grouped according to audit event services, audit service management, audit log management and audit log retrieval 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

- To handle event records newly generated at the local API level.
- The audit facility shall support the pre-selection of criteria for the detection of an event, thereby reducing the numbers of audit events generated and analysed.
- Filter and analyse 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 system 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.

1.1.2 Audit Service Management

These generic requirements are out of scope for the XDAS:

- 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 authorisations.
- Support both Remote and Local Administration
  The XDAS must support role-based decentralised 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.

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 the audit event 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.
- Record a security event message whenever a change to the configuration of the event discrimination service is made.

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 should have the ability to add domain specific information to the information set.

1.1.5 Audit Log Enquiry

The Audit Log Enquiry requirements are:

- Provide a common format definition for the audit log for use by analysis applications.

1.2 Security Requirements

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

- Prevent unauthorised modification of the audit service configuration data.
- Prevent unauthorised modification of the event detection records.
- Prevent unauthorised disclosure of the event records.
- Support adequate separation of duties for users.
- Provide appropriate measures in dealing with an unauthorised 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 unauthorised modification or deletions.
- Protect the audit log by making it accessible only to principals acting in specific administrative or security roles.

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

1.3 Distributed System Requirements

Two requirements need to be met by XDAS to support a distributed model. It must:

- Not hinder the achievement of adequate performance over the network.
- Utilise trustworthy universal timestamps on event records. Because the XDAS cannot assume a trusted time service is available, there is a requirement that the audit records 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.

1.4 Non-functional Requirements

The following non-functional requirements have been identified:

1. the XDAS shall be application independent
2. the XDAS shall not impose a particular placement of access control to distributed audit services within an operating system kernel
3. 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.

1.5 Out of Scope

The XDAS provides a set of primitives only, which are used by audit applications. 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 unauthorised modification of those event detection services.

Audit Filter Propogation

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 propogation 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 XDAS provides support for GUI tools. The specification supports 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

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 are required to 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.

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_initialise_session xdas_open_audit_stream
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 Conformamnce:

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

Chapter 3

XDAS Model

3.1 Introduction

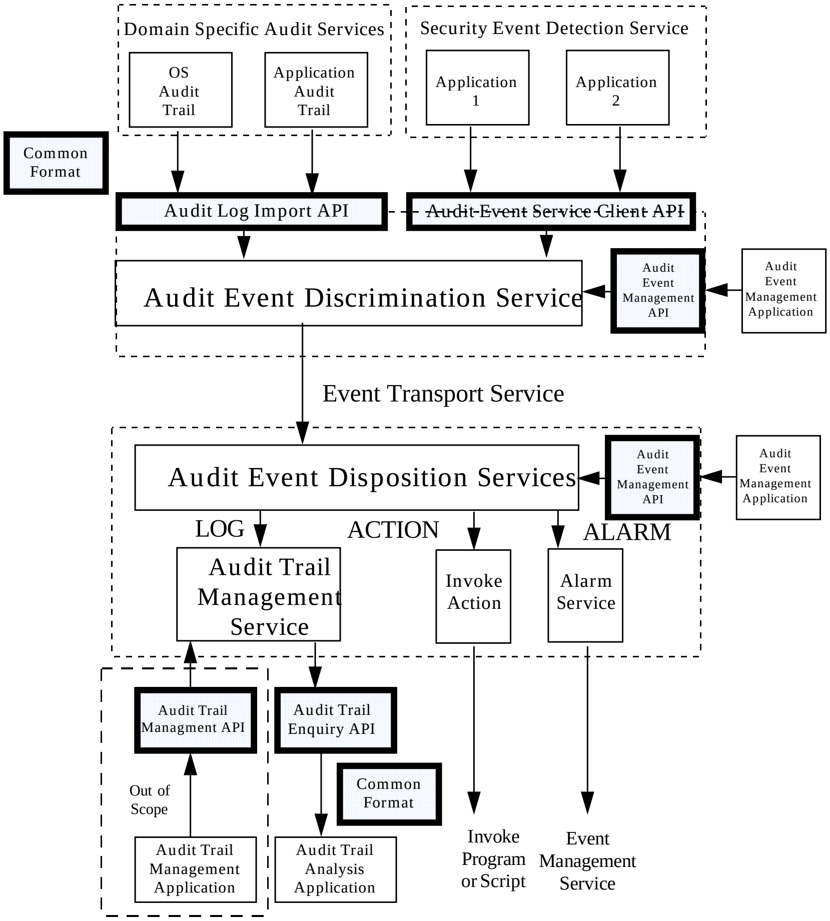
Domain Specific Audit Services
OS Audit Trail
Application Audit Trail
Security Event Detection Service
Application 1
Application 2
Common Format
Audit Log Import API
Audit Event Service Client API
Audit Event Discrimination Service
Audit Event Management API
Audit Event Management Application
Event Transport Service
Audit Event Disposition Services
Audit Event Management API
Audit Event Management Application
LOG
ACTION
ALARM
Audit Trail Management Service
Invoke Action
Alarm Service
Audit Trail Managment API
Audit Trail Enquiry API
Common Format
Out of Scope
Audit Trail Management Application
Audit Trail Analysis Application
Invoke Program or Script
Event Management Service

Figure 3-1 Distributed Audit Service Interfaces

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 3-1 on page 9. 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 local domain and to generate an audit event record which contains a description of the activity and information about the local security context. An application report 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 local 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.

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 *Audit Trail Management Service* supports 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 analysing 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 3-1 on page 9.

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 Service Client API

The *Audit Event Service Client 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 3-1 on page 9.

Audit Event Management API

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

Audit Trail Enquiry API

The *Audit Trail Enquiry API* is defined for the analysis 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 Model

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 X/Open 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.

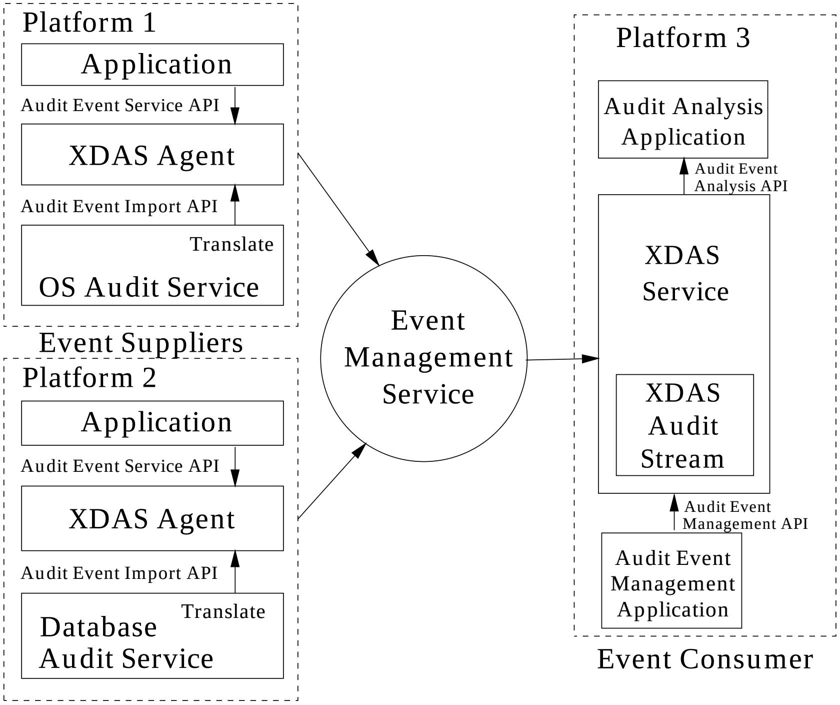

Figure 3-2 Distributed Audit Service Model

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 serviuce 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 organisation 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 ISO LATIN1 character set. This does not assume that the record contents are in a form that can be displayed as readable text. In addition, manifest constants should not be localised by any internationalisation 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. This 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 contain 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.)

  The XDAS specification includes the date and time of the start of the current EPOC which applies to the current version of the XDAS record format. (Start of the day January 1, 1970) Time is represented as the:

  - The *offset* in milliseconds from the beginning of the EPOC
  - The *uncertainty interval* in milliseconds of offset
  - The *uncertainty indicator* as a percentage of confidence in the uncertainty interval
  - The *signal* or *source* of trusted time.
  - The *timezone*

  The uncertainty interval and uncertainty indicator shall default to NULL. 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, ie., 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 initialised.

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 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* may include the 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. It is expected to comprise a number of *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 Appendix B.
- **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.
- **Authentication Authority**
  is defined using the syntax and quoting rules defined in Appendix B. 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 must be.

The *authentication authority*, *originator name* and *originator identity* represent the authenticated identity of the originator. Some of this information may not be available for inclusion in the audit record.

4.3.2 Initiator_Information

The information associated with an initiator comprises

- **Authentication Authority**
  defined using XFN syntax. 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 countries, for example, Germany, it is illegal to associate events 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.

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 *Initiator* (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.

The target of an activity that results in an auditable event is represented as for *originator_information*.

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 referenced by an *Event Number*. A set of Audit Events may be referenced by an *Event Class*. A potential 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 uses the event numbering scheme defined by the DCE auditing service in OSF RFC 29.2.

X/Open will register an Open Group set id and a set of numbers under that set id for the XDAS events identified. It is possible for application developers to register their own set of event numbers if they wish to utilise the services of XDAS for more domain specific auditing not catered for by the generic set of XDAS events

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 prinicipal 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**
  Request the attributes associated with a domain data itemt.

- **Modify data item attributes**
  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 Utilisation 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**
  Invocation of a service or application (exec), e.g., operating system utility, database, accounting application, etc.

- **Terminate service or application component**
  Terminate (exit) 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 adminsitrative action.

- **Query processing context**
  Query the attributes associated with the current processing environment.

- **Modify processing context**
  Modify 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 query 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**
  Receive data from associated peer within current association contxt.

- **Send data via an association**
  Send 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**
  Terminate an existing association with (close) a data item.

- **Query context of association with data item**
  Query 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**
  Modify the context of an association with a data item or resource element.

- **Query data item contents**
  Requesting the contents of a domain data item (read).

- **Modify data item contents**
  Modification of the contents of a domain data item (write, append etc).

Exceptional Events

These are events that are considered to be outside the generalised 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.

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

This could alternatively be called service exhaustion or service availability failure"

- **Resource corruption**
  The detection of an integrity failure of a system resource, for example data storage resource.

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

This could alternatively be called service integrity failure"

- **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 referenced 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 may 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_DENIED
	XDAS_OUT_PRE_SELECT_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
	XDAS_OUT_INSUFFICIENT_AUTHORISATION

Outcome	Outcome Description
	XDAS_OUT_INVALID_IDENTITY
	XDAS_OUT_INVALID_CREDENTIALS

4.5 Event Selection

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

Table 4-1 Event Filtering Criteria

Event selection may be applied at three places within the XDAS architecture:

- Pre-selection criteria may be applied when an event is detected to determine whether an event is to be logged or an action initiated, or both.
- 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 and if so whether the event is logged or an action initiated, or both.
- Post-selection criteria may be applied by audit analysis applications to control the selection of audit event records from the XDAS audit stream.

Table 4-1 on page 24 sets out the filtering criteria for pre-and post-selection criteria. An "X" indicates that the field is available for filtering; a "-" that it is not.

4.5.1 Event Submission Pre-Selection Filtering

The filtering criteria for pre-selection 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 Pre-Selection 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 Analysis Post Selection Filtering

Post selection filtering is the responsibility of the analysis application. XDAS does not itself apply filtering to the audit records returned by the Audit Read API and therefore does not include interfaces to support this.

4.5.4 Event Filters

An event filter comprises the following information:

Version Number
The XDAS version number.

Filter Name
A name by which the filter is referenced.

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 AND'd to establish the complete filter to be applied.

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

4.5.5 Filter Expression List

A filter expression list comprises a set of expressions that are ANDed to establish the complete expression to be applied. This specificiation 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 multiplle 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, bitwise ANDed, substring.

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

4.5.6 Filter Action List

The action list is a list comprising a constant to indicate the action and a text string.

XDAS_CONSTANT
The action to be taken. This can be LOG, ALARM or ACTION.

Text String
A text string that provides additional information pertinant to the action to be taken.

Examples of the *filter action list* are

- LOG + NULL 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 analyse 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 pre-selection 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 Authorisation Policy

The authorisation 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 authorisation security services. The following XDAS authorities have been defined.

XDAS_AUDIT_SERVICE
required to initialise 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

XDAS_AUDIT_PURGE
to authorise the removal of records from the XDAS audit stream

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

5.2 General Audit Service API

Initialise Session
Initialise 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 initialisation 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 initialise 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 initialised 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_ANALYSIS 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. Allocation of the **xdas_buffer_desc** object is always the responsibility of the application; unused **xdas_buffer_desc** objects may be initialised to the value XDAS_C_EMPTY_BUFFER.

6.3.2 Character Strings

Certain multi-octet data items may be regarded as simple Latin-1 character strings as defined in the ISO / IEC 8859-1 standard. 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 initialisation 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 Common Audit Record Format

The audit record format is defined as an ISO LATIN-1 character set in an **xdas_buffer_t** structure. Fields are delineated with colons (:); where a colon is part of the alphanumeric string, a % should be used as an escape character. Empty strings are represented by two adjacent separator characters. Note that this is an ordered sequence. The common audit record format is set out below:

field	**Type**
Header:	HDR
<length in bytes>	Digits 0-9
<ver#>	Digits 0-9
<date / time>	Hexadecimal
<offset>	Hexadecimal
<uncertainty interval>	Hexadecimal
<uncertainty indicator>	Hexadecimal
<time source>	Alphanumeric
<time zone>	Alphanumeric
<event_number>:	Hexadecimal
<outcome>	Hexadecimal
Originator	ORG
<location_name>	Alphanumeric
<location_address>	Alphanumeric
<service-type>	Alphanumeric
<auth_authority>	Alphanumeric
<principal_name>	Alphanumeric
<principal_id>	Alphanumeric
Initiator	INR
<auth_authority>	Alphanumeric
<domain_specific_name>	Alphanumeric
<domain_specific_id>	Alphanumeric
Target	TGT
<location_name>	Alphanumeric
<location_address>	Alphanumeric
<service-type>	Alphanumeric
<auth_authority>	Alphanumeric
<principal_name>	Alphanumeric
<principal_id>	Alphanumeric
Source	SRC
<pointer_to_source_domain>	Alphanumeric
Event	EVT
<event_specific_information>	Alphanumeric
END	END

The strings HDR, ORG, INR, TGT, SRC and EVT are included to support syntax checking. All fields should be included in the audit record, with separators, even if they are blank.

6.5 Filters

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

A filter comprises a name and a set of filter information. It is defined as:

```
typedef struct xdas_filter_desc_struct{
        xdas_buffer_t           filter_name;
        OM_unit32               filter_type;
        xdas_bool_t             flag;
        xdas_buffer_t           expression_list;
        xdas_buffer_t           action_list;
        } xdas_filter_desc, *xdas_filter_t;
```

A filter expression is defined as a **xdas_buffer_t** data type. It is a sequence of variable length ASCII Fields, separated by a ":" delimiter, as set out below. Note that if a colon is part of an alphanumeric string, the % should be used as an escape character. Empty strings are represented by two adjacent separator characters. The format for a filter expression is set out below:

field	**Type**
Include/Exclude Flag	Alphanumeric
Attribute	Alphanumeric
Operator	Alphanumeric
Value	alphanumeric

6.6 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.

6.6.1 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 erro from the routine and a single calling error. In addition, supplementary status information may be indicated by setting bits in a **Supplementary Info** field in a XDAS status code. These errors are encoded into the 32-bit XDAS status code as follows:

```
   MSB                                                        LSB
   |-----------------------------------------------------------|
   | Calling Error | Routine Error  |    Supplementary Info     |
   |-----------------------------------------------------------
Bit 31           24 23            16 15                       0
```

Hence if a XDAS-API routine returns a XDAS status code whose upper 16 bits contain 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 36. Whether or not the upper 16 bits indicate a failure or a success, the routine may indicate additional information by setting bits in the **Supplementary Info** field of the status code. This specification does not currently define any supplementary information but it is included to accommodate a possible future expansion in scope that might require such information.

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

Table 6-2 Routine Errors

Name	Value in Field	Meaning
[XDAS_S_COMPLETE]	0	Successful completion.
[XDAS_S_FAILURE]	1	An implementation specific error or failure has occurred.
[XDAS_S_AUTHORISATION_FAILURE]	2	The caller does not possess the required authority.
[XDAS_S_END]	3	The end of the audit stream has been reached.
[XDAS_S_INVALID_ACTION_LIST]	40	The action list supplied is not valid.
[XDAS_S_INVALID_AUDIT_STREAM]	5	The audit stream supplied is not valid.
[XDAS_S_INVALID_DAS_REF]	6	The audit daemon handle supplied does not point to the audit daemon.
[XDAS_S_INVALID_EVENT_INFO]	7	The specified audit event information is not valid.
[XDAS_S_INVALID_EVENT_NO]	8	The event number supplied is not valid.
[XDAS_S_INVALID_FILTER	9	The filter name supplied is not valid.
[XDAS_S_INVALID_FILTER_EXPR]	10	The filter expression supplied is not valid.
[XDAS_S_INVALID_FILTER_LIST]	11	The list of filter names supplied is not valid.
[XDAS_S_INVALID_FILTER_TYPE]	12	The filter type supplied is not valid.
[XDAS_S_INVALID_INITIATOR_INFO]	13	The initiator information has a syntax error.
[XDAS_S_INVALID_ORIG_INFO]	14	The originator information has a syntax error.
[XDAS_S_INVALID_OUTCOME]	15	The specified outcome is invalid.
[XDAS_S_INVALID_RECORD_DESCRIPTOR]	16	The specified audit record descriptor is not valid.
[XDAS_S_INVALID_SECURITY_CONTEXT]	17	The security context supplied is invalid.
[XDAS_S_INVALID_TARGET_INFO]	18	The target information has a syntax error.
[XDAS_S_NO_AUDIT]	19	The event does not need to be audited.
[XDAS_S_RECORD_SYNTAX_ERROR]	20	A syntax error has been detected in an input record
[XDAS_S_STORAGE_FAILURE]	21	The audit record cannot be written to stable storage.
[XDAS_S_SERVICE_FAILURE]	22	There has been an audit service failure
[XDAS_S_UNCERTAIN_AUDIT]	23	It is not certain whether the event should be audited.

The function specifications also use the name [XDAS_S_COMPLETE], which is a zero value, to indicate an absence of any API errors or supplementary information bits.

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_BAD_SIZE] (value 3 in the **Routine Error** field) is 3 << 16.

The macros:

XDAS_CALLING_ERROR( )
XDAS_ROUTINE_ERROR( )
XDAS_SUPPLEMENTARY_INFO( )

are 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** and **Supplementary Info** fields, 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.

A XDAS-API implementation may choose to signal calling errors in a platform-specific manner instead of, or in addition to th routine value; routine errors and supplementary information should be returned by means of routine status values only.

6.6.2 Minor Status Codes

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.

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 pointers (as in a **xdas_buffer_desc** structure) should also be set to zero in such cases. 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.

6.7 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.

6.7.1 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

6.7.2 Integer Types

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

6.7.3 Pointer Types

Specify NULL as the value.

6.8 Constants

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

Name	Value	Meaning
[XDAS_C_EMPTY_BUFFER]	NULL	Empty buffer
[XDAS_C_NO_BUFFER]	NULL	No buffer is supplied or returned.

Table 6-3 Optional Parameter Constants

Table 6-4 XDAS Event Field Separators

Separator	Purpose
HDR	Start of header data
ORG	Start of originator data
INR	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

6.9 Event Numbers

The following table defines the initial set of XDAS events numbers. These numbers will be converted into OpenGroup assigned numbers by addition to a root number once that number has been assigned.

Table 6-5 XDAS Event Numbers

Event Description	**Event Number**
Create account	1
Delete account	2
Disable account	3
Enable account	4
Query account attributes	5
Modify account attributes	6
Create a user session	7
Terminate a user session	8
Query a user session attributes	9
Modify user session attributes	10
Create data item	11
Delete data item	12
Query data item attributes	13
Modify data item attributes	14
Install service or application	15
Remove service or application	16
Query configuration of service or application	17
Modify configuration of service or application	18
Disable service or application	19
Enable service or application	20
Invoke service or application	21
Terminate service or application	22
Query processing context	23
Modify processing context	24
Create an association with a peer	25
Terminate an association with a peer	26
Query an association context	27
Modify an association context	28
Receive data via an association	29
Send data via an association	30
Create association with data item	31
Terminate association with data item	32
Query context of association with data item	33
Modify context of association with data item	34
Query data item contents	35
Modify data item contents	36
Start system	37
Shutdown system	38
Resource exhaustion	39

Event Description	**Event Number**
Resource corruption	40
Backup datastore	41
Recover datastore	42
Configure audit service	43
Audit datastore full	44
Audit datastore corrupted	45

6.10 XDAS Event Classes

The default set of event classes are:

Event Class Description	Event Class Code
Account management events	1
User session events	2
Data item and resource element management events	3
Service or application management events	4
Service and application utilisation events	5
Peer association management events	6
Data item or resource element content access events	7
Exceptional events	8
Audit service management events	9

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_PRE_SELECT_CRITERIA_SET]	"0x00000800"	Pre-selection criteria set or modified
[XDAS_OUT_THRESHOLDS_SET]	"0x00000800"	Thresholds set
[XDAS_OUT_ACTIONS_SET]	"0x00001000"	Actions set for alarms
[XDAS_OUT_THRESHOLD_EXCEEDED]	"0x00002000"	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_FAIURE]	"0x00000401"	Hardware failure or exception condition
[XDAS_OUT_LOST_ASSOCIATION]	"0x00000801"	Association lost
[XDAS_OUT_ALREADY_ENABLED]	"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 not valid

Table 6-7 XDAS Event Outcome Codes

6.12 XDAS Action Codes

The XDAS action codes 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]	3	Take specified action

Table 6-8 XDAS Action Codes

6.13 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
XDAS_C_ALL	3	All filters

Table 6-9 XDAS Filter Types

6.14 XDAS Filter Flags

The XDAS filter flags 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 6-10 XDAS Filter Flags

6.15 XDAS Filter Attributes

The XDAS filter attributes are:

Name	Value
XDAS_VERSION	1
XDAS_DATE_TIME	2
XDAS_EVENT_NUMBER	3
XDAS_OUTCOME	4
XDAS_ORG_LOC_NAME	5
XDAS_ORG_LOC_ADD	6
XDAS_ORG_SERV_TYPE	7
XDAS_ORG_AUTH_AUTH	8
XDAS_ORG_NAME	9
XDAS_ORG_IDENTITY	10
XDAS_INR_AUTH_AUTH	11
XDAS_INR_NAME	12
XDAS_INR_IDENTITY	13
XDAS_TRT_LOC_NAME	14
XDAS_TRT_LOC_ADD	15
XDAS_TRT_SERV_TYPE	16
XDAS_TRT_AUTH_AUTH	17
XDAS_TRT_NAME	18
XDAS_TRT_IDENTITY	19

Table 6-11 XDAS Filter Attributes

6.16 XDAS Filter Operators

The XDAS filter operators 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_BA	7	Bitwise AND
XDAS_O_SS	8	Substring

Table 6-12 XDAS Filter Operators

Chapter 7

XDAS Application Program Interface (API)

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

NAM

xdas_close_audit_stream — close the specified audit_stream

SYNOPSIS

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

DESCRIPTION

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

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

[XDAS_S_INVALID_AUDIT_STREAM]
The specified audit stream is not valid.

[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.

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

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
    xdas_audit_ref_t       *das_ref,
    xdas_audit_rec_desc_t         *audit_record_descriptor

);
```

DESCRIPTION

The XDAS implementation 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 caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE]. 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 XDAS service daemon, and the means by which the caller accesses the audit stream.

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

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

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

[XDAS_S_SERVICE_FAILURE]
There has been an audit service failure.

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

[XDAS_S_AUTHORISATION_FAILURE]
The caller does not possess the required authority

ERRORS

No other errors are defined.

NAME

xdas_create_filter — create the specified audit filter

SYNOPSIS

```
OM_uint32 das_create_filter (
    OM_uint32            *minor_status,
    xdas_audit_ref_t     *das_ref,
    xdas_buffer_t        *name,
    OM_unit32            *filter_type,
    xdas_buffer_t        *filter_exp,
    xdas_buffer_t        *filter_action_list,
);
```

DESCRIPTION

The *xdas_create_filter* function creates a filter for the *filter_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_initialise_session*.

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

name (optional,in)
: The name of the filter.

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

filter_action_list (optional,in) The list defining the actions to be taken on detecting the event.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
: Successful completion.

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

[XDAS_S_INVALID_FILTER]
: The filter name supplied already exists.

[XDAS_S_INVALID_FILTER_TYPE]
: The filter type supplied is not recognized.

[XDAS_S_INVALID_FILTER_EXP]
: The filter expression supplied is not valid.

[XDAS_S_INVALID_ACTION_LIST]
: The filter type supplied is not recognized.

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

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

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
    xdas_audit_ref_t      *das_ref,
    xdas_buffer_t         *name,
);
```

DESCRIPTION

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 deteletion 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_initialise_session*.

name (in)
The name of the filter.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_FILTER_TYPE]
The filter type supplied is not valid.

[XDAS_S_INVALID_FILTER]
The filter name supplied is not valid.

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

[XDAS_S_AUTHORISATION_FAILURE]
The caller does not possess the required authority

ERRORS

No other errors are defined.

NAME

xdas_disable_filter — disable the specified filter

SYNOPSIS

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

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 propogation 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:

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

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

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

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_FILTER]
The filter name supplied is not known.

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

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

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
     xdas_audit_ref_t             *das_ref,
     xdas_audit_desc_t            *audit_record_descriptor,
);
```

DESCRIPTION

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_initialise_session*.

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

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

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

[XDAS_S_AUTHORISATION_FAILURE]
The caller does not possess the required authority

ERRORS

No other errors are defined.

NAME

xdas_enable_filter — enable the specified audit filter

SYNOPSIS

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

DESCRIPTION

The *xdas_enable_filter* function enables the filter corresponding to the *name* specified. If necessary the enabled state of the filter may require propogation 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_initialise_session*.

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_FILTER]
The filter name supplied is not known.

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

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

ERRORS

No other errors are defined.

NAME

xdas_get_filter — get audit filters for a specified name

SYNOPSIS

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

DESCRIPTION

The *xdas_get_filter* function returns the components of the filter referenced 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_initialise_session*.

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

filter_type (in)
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_exp (out)
The contents of the expression list that determines the events to be 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_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_FILTER]
The filter name supplied is not known.

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

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

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
          xdas_audit_ref_t          *das_ref,
          xdas_audit_stream_t       *audit_stream_ref,
          OM_unit32                  max_records,
          xdas_buffer_t             *audit_record_buffer,
          OM_unit32                 *no_of_records,
);
```

DESCRIPTION

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*. The actual number of records copied 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 referenced by *das_ref* will be advanced to the next 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 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_initialise_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_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_STREAM_REF]
The audit stream handle supplied is invalid.

[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_AUTHORISATION_FAILURE]

The caller does not possess the required authority.

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
        xdas_audit_ref_t             das_ref,
        xdas_buffer_t               *audit_record_buffer,
        OM_uint32                    position_in_buffer,
);
```

DESCRIPTION

The *xdas_import_event_records* function allows a caller to submit 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 referenced by *audit-record_buffer* from which they are copied by XDAS and integrated into the XDAS audit stream. 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_initialise_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_COMPLETE]
: Successful completion.

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

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

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

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

ERRORS

No other errors are defined.

NAME

xdas_initialise_session — initialise a session with the distributed audit service

SYNOPSIS

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

DESCRIPTION

The *xdas_initialise_session* function initiates a session between the *server_identity* and the distributed audit service. It validates the *security_context* provided to ensure that caller has been authenticated and is authorised 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_initialise_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 (in)
An opaque structure containing 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 authorisations.

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_COMPLETE
Successful completion.

[XDAS_S_INVALID_SECURITY_CONTEXT]
The security context supplied is not valid.

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

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

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

ERRORS

No other errors are defined.

NAM

xdas_list_filters — list the audit filters that have been defined

SYNOPSIS

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

DESCRIPTION

The *xdas_list_filters* function returns a pointer to a NULL terminated list of the names of the filters that exist within the XDAS service. The caller must possess the XDAS_AUDIT_CONTROL authority.

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_initialise_session*.

filter_name_list (out)
: A pointer to 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_COMPLETE]
: Successful completion.

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

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

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

ERRORS

No other errors are defined

NAM

xdas_open_audit_stream — open the audit_stream

SYNOPSIS

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

DESCRIPTION

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_initialise_session*.

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

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 handle to the audit service is not valid.

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

ERROR

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,
    xdas_audit_ref_t      *das_ref,
    xdas_audit_desc_t            *audit_record_descriptor,
    OM_unit32             *event_number,
    OM_unit32             *outcome,
    xdas_buffer_t         *initiator_information,
    xdas_buffer_t         *target_information,
    xdas_buffer_t         *event_info
);
```

DESCRIPTION

The *xdas_put_event_info* function adds event specific information to an audit record. If the optional parameters are supplied, it also checks whether the specified event should be audited and returns an XDAS_AUDIT_UNCERTAIN or XDAS_NO_AUDIT code to the caller. Multiple calls to *xdas_put_event_info* may be made. For any individual parameter, information supplied in a call will overwrite any previous information supplied The order of the event information is preserved. The caller must have the XDAS_AUDIT_SUBMIT authority.

If successful, the function returns [XDAS_S_COMPLETE].

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_initialise_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. This is specified only if it has not already been set in the *audit_record_descriptor* supplied.

outcome (optional,in)
The outcome of the event determined by the caller. This is specified only if it has not already been set in the *audit_record_descriptor*
supplied.

initiator_information (optional,in)
The information describing the initiator in the format required by the XDAS common audit format. Again, this is optional, and is included only if it has not already been set in the *audit_record_descriptor* supplied.

target_information (optional,in)
The information on the target of the event in the format required by the XDAS common audit format. This is specified only if it has not already been set in the *audit_record_descriptor* supplied.

event_info (in)
The event specific information that is to be added to the audit record specified by

audit_record_descriptor.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

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

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

[XDAS_S_INVALID_EVENT_NO]
The specified event number is not valid.

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

[XDAS_S_UNCERTAIN_AUDIT]
There is uncertainty as to whether the event specified needs to be audited.

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

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

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

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
    xdas_audit_ref_t          *das_ref,
    xdas_buffer_t             *buffer,
);
```

DESCRIPTION

This function frees storage associated with a buffer. The storage must have been allocated by a 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_initialise_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 GCS status codes shall be returned:

[GCS_S_COMPLETE]
: Successful completion

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

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

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
    xdas_audit_ref_t             *das_ref,
    xdas_buffer_t                **filter_list,
);
```

DESCRIPTION

The *xdas_release_filter_list* function releases the list of filter names which were obtained by a previous call to *xdas_list_filters*. The caller must possess the XDAS_AUDIT_CONTROL authority.

If successful, the function returns [XDAS_S_COMPLETE].

The arguments for *xdas_release_filter_list*( ) are:

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

filter_list (in)
A pointer to the list of filter names obtained from a previous call to *xdas_list_filters*( ).

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

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

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

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

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

ERRORS

No other errors are defined.

xdas_rewind_audit_stream( )

NAME

xdas_rewind_audit_stream — rewind the audit stream

SYNOPSIS

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

DESCRIPTION

The *xdas_rewind_audit_stream* function rewinds the audit stream referenced 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_initialise_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_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_AUDIT_STREAM]
The specified audit stream is not valid.

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

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

ERRORS

No other errors are defined.

NAME

xdas_start_record — initialise an audit record

SYNOPSIS

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

DESCRIPTION

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 initialised 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 initiatior_informationsupplied. 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].

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_initialise_session*( ).

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

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.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

[XDAS_S_INVALID_EVENT_NO]
The event number specified is not valid.

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

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

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

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
    xdas_audit_ref_t       *das_ref,
);
```

DESCRIPTION

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_initialise_session*.

RETURN VALUE

The following XDAS status codes shall be returned:

[XDAS_S_COMPLETE]
Successful completion.

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

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

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

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
    xdas_audit_ref_t       *das_ref,
    xdas_audit_desc_t             *audit_record_descriptor,
);
```

DESCRIPTION

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_initialise_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_COMPLETE]
: Successful completion.

[XDAS_S_INVALID_DAS_REF]
: The audit daemon 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_FAILURE]
: An implementation specific error or failure has occurred.

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

ERRORS

No other errors are defined.

Appendix A

Mapping of DAS Events

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

These mappings need have been reworked using the revised set of XDAS events. The Oracle mappings are suspect as the editor is not familiar with the semantics of some of the statements.

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	create / delete data item
Alter / truncate cluster	modify data item
	Modify data item contents ??
Create / drop database link	create / delete data item
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

Oracle Event Description	XDAS-API Event(s)
Alter table, comment on table	modify data item
Execute procedure	invoke service or application
Grant/revoke privilege on procedure	configure service or application
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.1 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 GASAPI 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)	invoke service or application or enable service
msgctl(2)	modify data item attributes
msgget(2)	create data item, or create an association with peer
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

BSM Kernel-level Audit Events	XDAS-API Event
rename(2)	modify data item, modify data item attributes
rmdir(2)	delete data item
semctl(2)	modify data item attributes
semget(2)	create data item or create an association with peer
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	startup system components/services shutdown system components/services 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	creat an association with peer or create user session
/usr/bin/su	create user session or modify user session attributes

A.2 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
	modify data item contents
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	amodify data item attributes
	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
	modify data item contents
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

Appendix B

XDAS Naming Syntax

The XDAS name syntax is based upon the composite name syntax defined by the XFN Preliminary Specification. Although no divergance of syntax definition is planned, this specification will not necessarily be updated to reflect changes in the XFN specification as they may occur.

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

I have changed the reference to ISO 646 as used in XFN to ISO 8859-1.

B.1 Composite Name String Syntax

An *XDAS composite name* consists of an ordered list of zero or more components. 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. XFN does not specify any syntax for regular expressions at the composite name level. However, an individual naming system may allow a component to contain expressions (for example, wildcard characters).

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

B.1.1 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 ISO 8859-1 .

The minimum requirement for all XDAS implementations is to support the portable representation of ISO 8859-1 for communication of name strings.

B.1.2 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 XFN composite name syntax in BNF is as follows.

```
NULL ::=                // Empty set

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

Glossary

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

This glossary is the glossary from the XDSF. It needs those terms that are irrelevant to this specification removing and any other relevant terms adding.

access control

The prevention of unauthorised use of a resource including the prevention of use of a resource in an unauthorised manner (see).

access control certificate

ADI in the form of a security certificate (see).

access control decision function

(ADF) — a specialised function that makes access control decisions by applying access control policy rules to a requested action, ACI (of initiators, targets, actions, or that retained from prior actions), and the context in which the request is made (see).

access control decision information

(ADI) —) the portion (possibly all) of the ACI made available to the ADF in making a particular access control decision (see).

access control enforcement function

(AEF) — a specialised function that is part of the access path between an initiator and a target on each access that enforces the decisions made by the ADF (see).

access control information

(ACI) — any information used for access control purposes, including contextual information (see).

access control list

A list of entities, together with their access rights which are authorised to have access to a resource (see).

access control policy

The set of rules that define the conditions under which an access may take place (see).

accountability

The property that ensures that the actions of an entity may be traced to that entity (see).

ACI

Access control information.

ACL

Access control list.

action

The operations and operands that form part of an attempted access (see).

action ADI

Action decision information associated with the action (see).

active threat

The threat of a deliberate unauthorised change to the state of the system

ADF

Access control decision function.

ADI

Access control decision information.

administrative security information

Persistent information associated with entities; it is conceptually stored in the Security Management Information Base. Examples are:

- security attributes associated with users and set up on user account installation, which is used to configure the user's identity and privileges within the system
- information configuring a secure interaction policy between one entity and another entity, which is used as the basis for the establishment of operational associations between those two entities.

AEF

Access control enforcement function.

alarm collector function

A function that collects the security alarm messages, translates them into security alarm records, and writes them to the security alarm log (see).

alarm examiner function

A function that interfaces with a security alarm administrator (see).

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 **Procurement Guide**).

assertion

Explicit statement in a system security policy that security measures in one security domain constitute an adequate basis for security measures (or lack of them) in another (see).

association-security-state

The collection of information that is relevant to the control of communications security for a particular application-association (see).

audit

See Security Audit (see).

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 authority

The manager responsible for defining those aspects of a security policy applicable to maintaining a security audit (see).

audit event detector function

A function that detects the occurrence of security-relevant events. This function is normally an inherent part of the functionality implementing the event (see).

audit event discriminator function

A function that filters audit events against pre-configured criteria. The filter mechanism is parameter driven, based on policies or rules. This function may be invoked prior to event generation, to determine whether a detected audit event is required to be audited, or after event generation to determine how a generated event is to be handled, for example logged or an alarm generated.

audit recorder function

A function that records the security-relevant messages in a security audit trail (see).

audit trail

See Security Audit Trail (see).

audit trail analyser function

A function that checks a security audit trail in order to produce, if appropriate, security alarm messages (see).

audit trail archiver function

A function that archives a part of the security audit trail (see).

audit trail collector function

A function that collects individual audit trail records into a security audit trail (see).

audit trail examiner function

A function that builds security reports out of one or more security audit trails (see).

audit trail provider function

A function that provides security audit trails according to some criteria (see).

authenticated identity

An identity of a principal that has been assured through authentication (see).

authentication

Verify claimed identity; see data origin authentication, and peer entity authentication (see).

authentication certificate

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).

authentication exchange

A sequence of one or more transfers of exchange authentication information (AI) for the purposes of performing an authentication (see).

authentication information (AI)

Information used to establish the validity of a claimed identity (see).

authentication initiator

The entity which starts an authentication exchange (see).

authentication method

Method for demonstrating knowledge of a secret. The quality of the authentication method, its strength is determined by the cryptographic basis of the key distribution service on which it is based. A symmetric key based method, in which both entities share common authentication information, is considered to be a weaker method than an asymmetric key based method, in which not all the authentication information is shared by both entities.

authorisation

The granting of rights, which includes the granting of access based on access rights (see).

authorisation 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 authorisation policy may be defined in terms of access control lists, capabilities or attributes assigned to security subjects, security objects or both (see).

availability

The property of being accessible and usable upon demand by an authorised entity (see).

capability

A token used as an identifier for a resource such that possession of the token confers access rights for the resource (see).

ciphertext

Data produced through the use of encipherment. The semantic content of the resulting data is not available (see).

Note: Ciphertext may itself be input to encipherment, such that super-enciphered output is produced.

claim authentication information

(Claim AI) — information used by a claimant to generate exchange AI needed to authenticate a principal (see).

claimant

An entity which is or represents a principal for the purposes of authentication. A claimant includes the functions necessary for engaging in authentication exchanges on behalf of a principal (see).

clear text

Intelligible data, the semantic content of which is available (see).

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 unauthorised individuals, entities, or processes (see).

contextual information

Information derived from the context in which an access is made (for example, time of day) (see).

corporate security policy

The set of laws, rules and practices that regulate how assets including sensitive information are managed, protected and distributed within a user organisation (see).

countermeasure

The deployment of a set of security services to protect against a security threat.

credentials

Data that is transferred to establish the claimed identity of an entity (see).

cryptanalysis

The analysis of a cryptographic system and its inputs and outputs to derive confidential variables and/or sensitive data including clear text (see).

cryptographic algorithm

A method of performing a cryptographic transformation (see cryptography) on a data unit. Cryptographic algorithms may be based on symmetric key methods (the same key is used for both encipher and decipher transformations) or on asymmetric keys (different keys are used for encipher and decipher transformations).

cryptographic checkvalue

Information that is derived by performing a cryptographic transformation (see cryptography) on a data unit (see).

Note: The derivation of the checkvalue may be performed in one or more steps and is a result of a mathematical function of the key and data unit. It is usually used to check the integrity of a data unit.

cryptography

The discipline that embodies principles, means, and the methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use (see).

Note: The choice of cryptography mechanism determines the methods used in encipherment and decipherment. An attack on a cryptographic principle, means or methods is cryptanalysis.

data integrity

The property that data has not been altered or destroyed in an unauthorised manner (see).

data origin authentication

The corroboration that the entity responsible for the creation of a set of data is the one claimed.

decipherment

The reversal of a corresponding reversible encipherment (see).

decryption

See decipherment (see).

denial of service

The unauthorised prevention of authorised access to resources or the delaying of time-critical operations (see).

digital fingerprint

A characteristic of a data item, such as a cryptographic checkvalue or the result of performing a one-way hash function on the data, that is sufficiently peculiar to the data item that it is computationally infeasible to find another data item that possesses the same characteristics (see).

digital signature

Data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery for example, by the recipient (see).

discretionary access control

A discretionary authorisation scheme is one under which any principal using the domain services may be authorised to assign or modify ACI such that he may modify the authorisations of other principals under the scheme. A typical example is an ACL scheme which is often referred to as Discretionary Access Control (DAC).

distinguishing identifier

Data that unambiguously distinguishes an entity in the authentication process. Such an identifier shall be unambiguous at least within a security domain (see).

distributed application

A set of information processing resources distributed over one or more open systems which provides a well-defined set of functionality to (human) users, to assist a given (office) task (see).

encapsulated subsystem

A collection of procedures and data objects that is protected in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the encapsulated subsystem and that those procedures may be called only at designated domain entry points. Encapsulated subsystem, protected subsystem and protected mechanisms of the TCB are terms that may be used interchangeably (see).

encipherment

The cryptographic transformation of data (see cryptography) to produce ciphertext (see).

Note: Encipherment may be irreversible, in which case the corresponding decipherment process cannot feasibly be performed. Such encipherment may be called a one-way-function or cryptochecksum.

encryption

See encipherment (see).

end-to-end encipherment

Encipherment of data within or at the source end system, with the corresponding decipherment occurring only within or at the destination end system (see).

exchange authentication information

(Exchange AI) — information exchanged between a claimant and a verifier during the process of authenticating a principal (see).

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.

identity-based security policy

A security policy based on the identities or attributes of users, a group of users, or entities acting on behalf of the users and the resources or targets being accessed (see).

initiator

An entity (for example, human user or computer based entity) that attempts to access other entities (see).

initiator access control decision information

(Initiator ADI) — ADI associated with the initiator (see).

initiator access control information

(Initiator ACI) — access control information relating to the initiator (see).

integrity

See Data Integrity (see).

key

A sequence of symbols that controls the operations of encipherment and decipherment (see).

key management

The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy (see).

masquerade

The unauthorised pretence by an entity to be a different entity (see).

messaging application

An application based on a store and forward paradigm; it requires an appropriate security context to be bound with the message itself.

non-discretionary access control

A non-discretionary authorisation scheme is one under which only the recognised security authority of the security domain may assign or modify the ACI for the authorisation scheme such that the authorisations of principals under the scheme are modified.

off-line authentication certificate

A particular form of authentication information binding an entity to a cryptographic key, certified by a trusted authority, which may be used for authentication without directly interacting with the authority (see).

on-line authentication certificate

A particular form of authentication information, certified by a trusted authority, which may be used for authentication following direct interaction with the authority (see).

operational security information

Transient information related to a single operation or set of operations within the context of an operational association, for example, a user session. Operational security information represents the current security context of the operations and may be passed as parameters to the operational primitives or retrieved from the operations environment as defaults.

organisational security policy

Set of laws, rules, and practices that regulates how an organisation manages, protects, and distributes sensitive information (see).

password

Confidential authentication information, usually composed of a string of characters (see).

peer-entity authentication

The corroboration that a peer entity in an association is the one claimed (see).

physical security

The measures used to provide physical protection of resources against deliberate and accidental threats (see).

platform domain

A security domain encompassing the operating system, the entities and operations it supports and its security policy.

policy

See security policy (see).

primary service

An independent category of service such as operating system services, communication services and data management services. Each primary service provides a discrete set of functionality. Each primary service inherently includes generic qualities such as usability, manageability and security.

Security services are therefore not primary services but are invoked as part of the provision of primary services by the primary service provider.

principal

An entity whose identity can be authenticated (see).

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).

private key

A key used in an asymmetric algorithm. Possession of this key is restricted, usually to only one entity (see).

public key

The key, used in an asymmetric algorithm, that is publicly available (see).

quality of protection

A label that implies methods of security protection under a security policy. This normally includes a combination of integrity and confidentiality requirements and is typically implemented in a communications environment by a combination of cryptographic mechanisms.

repudiation

Denial by one of the entities involved in a communication of having participated in all or part of the communication (see).

rule-based security policy

A security policy based on global rules imposed for all users. These rules usually rely on a comparison of the sensitivity of the resources being accessed and the possession of corresponding attributes of users, a group of users, or entities acting on behalf of users (see).

seal

A cryptographic checkvalue that supports integrity but does not protect against forgery by the recipient (that is, it does not support non-repudiation). When a seal is associated with a data element, that data element is *sealed* (see).

secondary discretionary disclosure

An example of the misuse of access rights. It occurs when a principal authorised to access some information copies that information and authorises access to the copy by a second principal who is not authorised to access the original information.

secret key

In a symmetric cryptographic algorithm the key shared between two entities (see).

secure association

An instance of secure communication (using communication in the broad sense of space and/or time) which makes use of a secure context.

secure context

The existence of the necessary information for the correct operation of the security mechanisms at the appropriate place and time.

secure interaction policy

The common aspects of the security policies in effect at each of the communicating application processes (see).

security architecture

A high level description of the structure of a system, with security functions assigned to components within this structure (see).

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).

security audit message

A message generated following the occurrence of an auditable security-related event (see).

security audit record

A single record in a security audit trail corresponding to a single security-related event (see).

security audit trail

Data collected and potentially used to facilitate a security audit (see).

security auditor

An individual or a process allowed to have access to the security audit trail and to build audit reports (see).

security aware

The caller of an API that is aware of the security functionality and parameters which may be provided by an API.

security certificate

A set of security-relevant data from an issuing security authority that is protected by integrity and data origin authentication, and includes an indication of a time period of validity (see).

Note: All certificates are deemed to be security certificates (see the relevant definitions in) adopted in order to avoid terminology conflicts with (that is the directory authentication standard).

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).

security event manager

An individual or process allowed to specify and manage the events which may generate a security message and to establish the action or actions to be taken for each security message type (see).

security label

The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource (see).

Note: The marking may be explicit or implicit.

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.

security state

State information that is held in an open system and which is required for the provision of security services.

security token

A set of security-relevant data that is protected by integrity and data origin authentication from a source that is not considered a security authority (see).

security unaware

The caller of an API that is unaware of the security functionality and parameters which may be provided by an API.

sensitivity

The characteristic of a resource that implies its value or importance, and may include its vulnerability (see).

separation

The concept of keeping information of different security classes apart in a system (see).

Note: Separation may be implemented by temporal, physical, logical or cryptographic techniques.

service domain

A security domain encompassing an application, the entities and operations it supports and its security policy.

signature

See digital signature (see).

strength of mechanism

An aspect of the assessment of the effectiveness of a security mechanism, namely the ability of the security mechanism to withstand direct attack against deficiencies in its underlying algorithms, principles and properties (see).

system security function

A capability of an open system to perform security-related processing (see).

target

An entity to which access may be attempted (see).

target ADI

ADI associated with the target (see).

target ACI

Access control information relating to the target (see).

threat

A potential violation of security (see).

An action or event that might prejudice security (see).

traffic analysis

The inference of information from observation of traffic flows (presence, absence, amount, direction and frequency) (see).

traffic flow confidentiality

A confidentiality service to protect against traffic analysis (see).

traffic padding

The generation of spurious instances of communication, spurious data units or spurious data within data units (see).

trap door

A hidden software or hardware mechanism that permits system protection mechanisms to be circumvented. It is activated in some non-apparent manner (for example, special ‘‘random’’ key sequence at a terminal) (see).

trojan horse

Computer program containing an apparent or actual useful function that contains additional (hidden) functions that allow unauthorised collection, falsification or destruction of data (see).

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).

trusted computing base (TCB)

The totality of protection mechanisms within an IT system, including hardware, firmware, software and data, the combination of which is responsible for enforcing the security policy.

trusted functionality

That which is perceived to be correct with respect to some criteria, for example, as established by a security policy (see).

trusted path

Mechanism by which a person using a terminal can communicate directly with the TCB (see).

Note: Trusted path can only be activated by the person or the TCB and cannot be imitated by untrusted software.

trusted third party

A security authority or its agent, trusted by other entities with respect to security-related operations (see).

verification AI

Information used by a verifier to verify an identity claimed through exchange AI (see).

verifier

An entity which is or represents the entity requiring an authenticated identity. A verifier includes the functions necessary for engaging in authentication exchanges (see).

virus

Self replicating, malicious program segment that attaches itself to an application or other executable system component and leaves no external signs of its presence (see).

vulnerability

Weakness in an information system or components (for example, system security procedures, hardware design, internal controls) that could be exploited to produce an information-related misfortune (see).

Index