# Authorization Token Layer Acquisition Service (ATLAS) Specification

**October 2002**
**Version 1.0**
**formal/02-10-01**

**An Adopted Specification of the Object Management Group, Inc.**

## USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

## LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

## PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

## GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

## DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.

IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

## RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

## TRADEMARKS

The OMG Object Management Group Logo®, CORBA®, CORBA Academy®, The Information Brokerage®, XMI® and IIOP® are registered trademarks of the Object Management Group. OMG™, Object Management Group™, CORBA logos™, OMG Interface Definition Language (IDL)™, The Architecture of Choice for a Changing World™, CORBAservices™, CORBAfacilities™, CORBAmed™, CORBAnet™, Integrate 2002™, Middleware That's Everywhere™, UML™, Unified Modeling Language™, The UML Cube logo™, MOF™, CWM™, The CWM Logo™, Model Driven Architecture™, Model Driven Architecture Logos™, MDA™, OMG Model Driven Architecture™, OMG MDA™ and the XMI Logo™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

## COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

## ISSUE REPORTING

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents & Specifications, Report a Bug/Issue.

# *Contents*

# *Preface*

## *About This Document*

Under the terms of the collaboration between OMG and The Open Group, this document is a candidate for adoption by The Open Group, as an Open Group Technical Standard. The collaboration between OMG and The Open Group ensures joint review and cohesive support for emerging object-based specifications.

### *Object Management Group*

The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common framework for application development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heterogeneous environments. Conformance to these specifications will make it possible to develop a heterogeneous applications environment across all major hardware platforms and operating systems.

OMG's objectives are to foster the growth of object technology and influence its direction by establishing the Object Management Architecture (OMA). The OMA provides the conceptual infrastructure upon which all OMG specifications are based. More information is available at http://www.omg.org/.

### *The Open Group*

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

The mission of The Open Group is to drive the creation of boundaryless information flow achieved by:

- Working with customers to capture, understand and address current and emerging requirements, establish policies, and share best practices;
- Working with suppliers, consortia and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies;
- Offering a comprehensive set of services to enhance the operational efficiency of consortia; and
- Developing and operating the industry's premier certification service and encouraging procurement of certified products.

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

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

## *OMG Documents*

The OMG Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

The OMG documentation is organized as follows:

### *OMG Modeling Specifications*

Includes the UML, MOF, XMI, and CWM specifications.

### *OMG Middleware Specifications*

Includes CORBA/IIOP, IDL/Language Mappings, Specialized CORBA specifications, and CORBA Component Model (CCM).

### *Platform Specific Model and Interface Specifications*

Includes CORBAservices, CORBAfacilities, OMG Domain specifications, OMG Embedded Intelligence specifications, and OMG Security specifications.

### *Obtaining OMG Documents*

The OMG collects information for each book in the documentation set by issuing Requests for Information, Requests for Proposals, and Requests for Comment and, with its membership, evaluating the responses. Specifications are adopted as standards only when representatives of the OMG membership accept them as such by vote. (The policies and procedures of the OMG are described in detail in the *Object Management Architecture Guide*.) OMG formal documents are available from our web site in PostScript and PDF format. Contact the Object Management Group, Inc. at:


OMG Headquarters

250 First Avenue

Needham, MA 02494

USA

Tel: +1-781-444-0404

Fax: +1-781-444-0320

pubs@omg.org

http://www.omg.org


## *Typographical Conventions*

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

**Helvetica bold** - OMG Interface Definition Language (OMG IDL) and syntax elements.

**`Courier bold`** - Programming language elements.

Helvetica - Exceptions

Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

## *Acknowledgments*


The following companies submitted and/or supported parts of this specification:

- Adiron, LLC
- Compaq Computer Corporation
- Concept Five Technologies, Inc.
- Promia, Inc.
- Syracuse University

# Overview 1

## Contents

This chapter contains the following topics.



## 1.1 Introduction

This chapter discusses some of the design decisions that were made to provide the components to achieve full secure interoperability between clients and targets.

The ATLAS specification describes the service needed to acquire authorization tokens to access a target system using the newly adopted CSIv2 protocol. This design, mandated by the RFP, defines a single interface with which to a client acquires an authorization token for a particular token. This token may be pushed, using the CSIv2 protocol in order to gain access to a CORBA invocation on the target.

This specification solves the problem of acquiring the privileges needed for a client to acquire a set of privileges the target will understand as the client need not understand the token that is retrieved from the ATLAS.

## 1.2 Relationship to Other OMG Modules

This document refers to the following modules:

- module CSI

- module CSIIOP
- module Time
- module CosNaming
- module CosNamingExt

## 1.3 Existing Specifications

This section describes the relationships of this specification with other existing CORBA specifications.

### 1.3.1 Use of Existing Specifications

This specification is dependent on the ORB services, data structures, and semantic definitions defined in the following specifications:

- OMG Naming Service (OMG TC Document orbos/99-10-11) [2]
- OMG Time Service (OMG TC Document formal/2000-06-26) [3]
- OMG Common Secure Interoperability Version 2 RFP Response (OMG TC Document orbos/2000-08-04)

#### 1.3.1.1 The Name Service

Dependencies on the Naming Service specification include:

- use of the Name Services **CosNaming::NamingContext** interface,
- use of the definition of **CosNaming::NamingContextExt::StringName**, and
- use of the definition of **CosNaming::NamingContextExt::URLString**.

#### 1.3.1.2 The Time Service

Dependencies on the Time Service specification are limited to the **TimeBase::UtcT** database structure.

#### 1.3.1.3 The CSIv2 Interoperability Specification

Dependencies on the CSIv2 specification include:

- use of the **CSI::IdentityToken**,
- use of the **CSI::AuthorizationToken**,
- use of the **CSIIOP::ServiceConfiguration**

structures, and their related types.

# ATLAS Specification 2

## Contents

This chapter contains the following topics.



## 2.1 Introduction

This section describes the Authorization Token Layer Acquisition Service (ATLAS). The Authorization Token Acquisition Service is a service by which a client’s security service (CSS) acquires authorization tokens to deliver to a target’s security service (TSS).

An authorization token consists of information that is processed by a TSS for security purposes. For example, the TSS uses the information in the authorization token to grant or deny access to the target's resources on behalf of the client.

Authorization tokens must contain privilege information that is scoped to the target's understanding of privileges to be effective. Privilege information must be scoped to the target's realm of understood privileges. For example, the privilege "doctor" at target Hospital A does not necessarily have the same meaning as "doctor" at target Hospital B, if any meaning at all. Alternatively, a single client, such as "Alice," may have the "doctor" privilege at Hospital A, but not at Hospital B.

Privileges are defined by privilege authorities, which define the privilege scope. The previous example illustrates that two different entities define the privilege "doctor," as well as the mappings from clients to those privileges. Hospital A subscribes to one privilege authority, Hospital B subscribes to a different privilege authority. The hospitals have different privilege scopes.

The CSS needs to deliver an authorization token that is within the target's privilege scope. The definition of a privilege scope consists of, but is not limited to, the following capabilities:

- Authorizing the client with privileges defined by a privilege authority that is understood by the target.
- Authorizing the target to be endorsed with the client's privileges or identity should it be necessary for the needs of both the client and target.

To facilitate secure interoperability, the TSS indicates to a client the location of the specific ATLAS that defines the target's privilege scope. The CSS retrieves from that ATLAS an authorization token, and the CSS is guaranteed that the token is understood by the TSS.

Many different targets may belong to the same privilege scope and therefore they may indicate the use of the same ATLAS. One client may use many of these targets. This specification defines the data structures and semantics with which TSS's convey the location of the target's ATLAS. The approach defined in this specification facilitates client caching of the authorization tokens based on the privilege scope.

## 2.2 Specification Scope

An ATLAS only delivers authorization tokens for one privilege scope. A service that issues authorization tokens of a variety of different token formats and different scopes is outside of this specification.

This specification only addresses retrieval of authorization tokens that clients deliver to targets for security purposes. Specification of the delivery of those tokens from the CSS to the TSS is left to a transport mechanism and is outside the scope of this specification. Also, definition of the mechanism by which the TSS transmits the location of its ATLAS to the CSS is left to a transport mechanism and is also outside the scope of this specification.

Administration of privileges, privilege scopes, and token formats is outside the scope of this specification.

This specification facilitates the notion of client caching of authorization tokens. This specification defines the caching semantics. However, the specific caching mechanism and the conditions on which the client chooses to cache authorization tokens is outside the scope of this specification.

This specification defines the method by which a client locates an ATLAS, but only does so with respect to making an interoperable request on a specific target. The client may know beforehand the various ATLAS's in which it will be dealing. In that case, it may want to locate thoseATLAS's and fill its cache with frequently used authorization tokens. Locating those ATLAS's for this purpose is outside the scope of this specification. However, it might be helpful to mention to the implementers that naming or trading services may be employed to do so.

## 2.3 Reference Model for CSIv2 Authorization Interoperability

The following model illustrates authorization interoperability with the CSIv2 authorization layer that this specification supports.

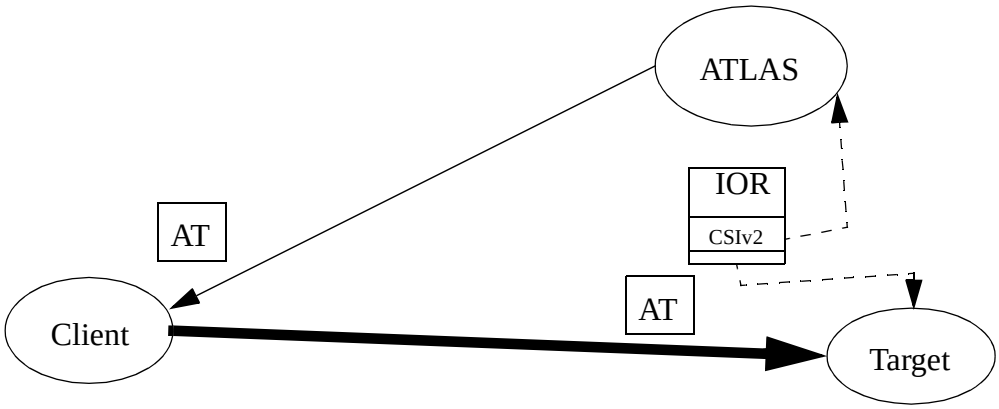


*Figure 2-1* Reference model for CSIv2 authorization interoperability

In the above model, the Client has no prior agreements with the Target. The Client has no knowledge as to the format and authority of an Authorization Token (AT) that the Target will understand. The following scenario ensues:

1. The Client acquires the IOR of the Target.
2. The Client looks at the CSIv2 component in the Target's IOR and locates the ATLAS based on the information in the **CSIIOP::SAS_ContextSec:privilege_authorities** field.
3. The Client requests an Authorization Token from the Target's ATLAS based on its own authentication to the ATLAS.

4. The Client makes its intended CSIv2 protected invocation on the Target pushing the AT that was retrieved from the ATLAS in the authorization layer of the CSIv2 protocol.

5. Since the Target specified the ATLAS, the Target will understand the format and encoding of the AT that is produced by the ATLAS. The Client need not understand the format or encoding of the AT.

## 2.4 Reference Model for Spanning Authorization Domains

The following model illustrates authorization interoperability with the CSIv2 authorization layer when Authorization Domains may be crossed.

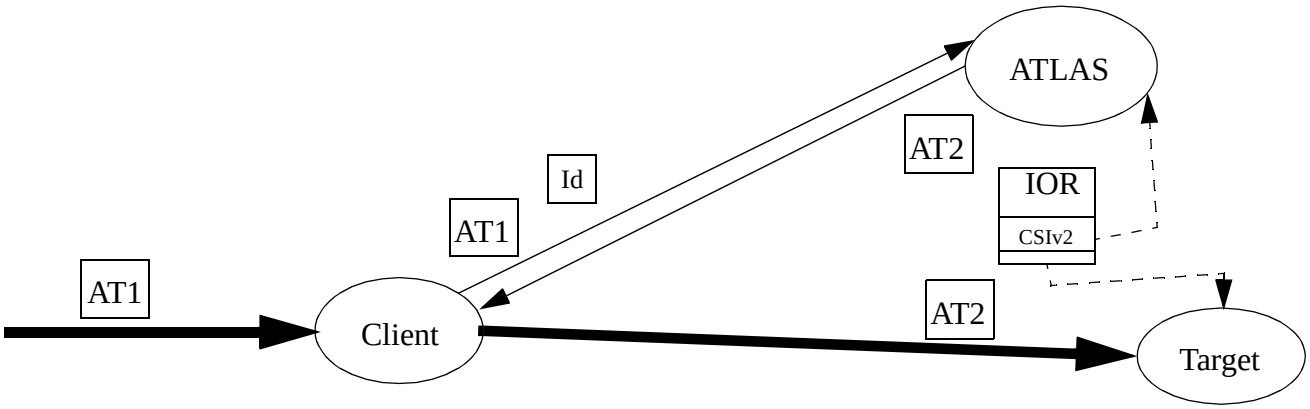


*Figure 2-2* Reference model for spanning authorization domains

In the above model, the Intermediary Client has no prior agreements with the Target. The Client has no knowledge as to the authorization domain, and the format and encoding of an Authorization Token that the Target will understand. However, the Client has an Authorization Token, AT1, from another Authorization Domain. The following scenario ensues:

1. The Client acquires the IOR of the Target.

2. The Client looks at the CSIv2 component in the Target's IOR and locates the ATLAS based on the information in the **CSIIOP::SAS_ContextSec:privilege_authorities** field.

3. The Client intends to make a request on the Target in some other principal's behalf. The Client requests AT2 from the ATLAS based on its own authentication to the ATLAS, the other principal's authorization token (AT1), and the identity token representing the other principal.

4. The Client makes its intended CSIv2 protected invocation on the Target pushing AT2 that was retrieved from the ATLAS in the authorization layer of the CSIv2 protocol and asserting the identity of the other principal.

Since the Target specified the ATLAS, the Target will understand the format and encoding of the AT2 that is produced by the ATLAS. The Client need not understand the format or encoding of the AT2.

## 2.5 The ATLAS Module

The ATLAS module contains data types, exceptions, and interfaces used by anATLAS. Some of the important types are described here. The ATLAS module contains the following IDL:

```
// File: ATLAS.idl

#ifndef _ATLAS_IDL_
#define _ATLAS_IDL_

#include <TimeBase.idl>
#include <CosNaming.idl>
#include <CSI.idl>
#include <CSIIOP.idl>

#pragma prefix "omg.org"

module ATLAS {

...

};
#endif // _ATLAS_IDL_
```

The ATLAS module depends on some data types in the **TimeBase**, **CosNaming**, **CSI**, and **CSIIOP** modules. Some important types are described in the following sections. The full IDL description of the module is defined in Appendix C, “OMG IDL.”

### 2.5.1 The ExpiryTime Type

The **ExpiryTime** structure is a component of an **AuthTokenData** structure that stipulates the time that authorization token will expire, if known. The **ExpiryTime** is a sequence of at most one element of coordinated universal time. A zero element sequence indicates that the expiry time of the authorization token is not known. The **ExpiryTime** type has the following definition:

```
typedef sequence<TimeBase::UtcT,1> ExpiryTime;
```

### 2.5.2 The IdTokenOption Type

The CSIv2 protocol requires the use of a **CSI::IdentityToken**. The intended identity shall be asserted along with the **CSI::AuthorizationToken** to give the TSS an indication of the identity to which the **CSI::AuthorizationToken** pertains. The return of a **CSI::IdentityToken** also facilitates translation of the identity token as different privilege scopes may map identities to different encodings, or even different identities.

The **IdTokenOption** structure is a component of an **AuthTokenData** structure that stipulates the **CSI::IdentityToken** that the CSS shall use in conjunction with the **CSI::AuthorizationToken**. The **IdTokenOption** is a sequence of at most one element containing a **CSI::IdentityToken**. A zero element sequence indicates that the **CSI::IdentityToken** used by the CSS in its call to the ATLAS is acceptable in conjunction with the accompanying **CSI::AuthorizationToken**. This approach removes the need to return the same token back to the CSS, as the identity token can be lengthy.

```
typedef sequence<CSI::IdentityToken,1> IdTokenOption;
```

### 2.5.3 The AuthTokenData Type

The **AuthTokenData** structure is used for a return value in some ATLAS operations. It returns the **CSI::IdentityToken**, **CSI::AuthorizationToken**, and an expiry time. The expiry time shall indicate the time the token will expire, if known. It has the following definition:

```
struct AuthTokenData {
    IdTokenOption               ident_token;
    CSI::AuthorizationToken     auth_token;
    ExpiryTime                  expiry_time;
};
```

The **CSI::AuthorizationToken** type is actually a sequence of **CSI::AuthorizationElement**. Therefore, the **auth_token** field may be a sequence that contains zero elements, which means that the authorization token is empty. In this case, the CSS shall send an empty authorization token to the intended TSS.

### 2.5.4 The AuthTokenDispenser Interface

The **AuthTokenDispenser** interface delivers **AuthTokenData** elements to the client. It has the following definition:

```
interface AuthTokenDispenser {
    // ... attributes and operations
};
```

The operations of the **AuthTokenDispenser** interface are defined in the following subsections.

### 2.5.4.1 *get_my_authorization_token*

A client shall use this operation to retrieve an authorization token based on the client's own identity. It has the following definition:

```
AuthTokenData get_my_authorization_token()
    raises (
        IllegalTokenRequest
);
```

***Return Value***

The value returned by this operation shall be the data structure containing the **CSI::AuthorizationToken**, and **CSI::IdentityToken** for the client. An **IllegalTokenRequest** exception shall be raised in the event that the client is not granted an authorization token.

### 2.5.4.2 *translate_authorization_token*

A client shall use this operation to translate an authorization token from one privilege scope to that of the scope supported by this ATLAS for the intended subject. It has the following definition:

```
AuthTokenData translate_authorization_token(
    in  CSI::IdentityToken      the_subject,
    in  AuthorizationToken      the_token
) raises (
    IllegalTokenRequest,
    TokenOkay
);
```

The client may use this operation to "request" privileges within the target scope by creating an authorization token and having it translated.

***Parameters***

| | |
|---|---|
| *the_subject* | This parameter specifies the identity for which the token is being translated. The **CSI::IdentityToken** type is a discriminated union that accommodates different name forms. A client shall not use a **CSI::IdentityToken** with a discriminator of **CSI::ITTAbsent**. |
| *the_token* | This parameter contains the token to be translated. |

***Return Value***

The value returned shall be the structure containing the **CSI::AuthorizationToken** and the **CSI::IdentityToken** that has been translated to the target's privilege scope.

An **IllegalTokenRequest** exception shall be raised in the event that the client is not granted an authorization token, or if the given authorization token is not translatable by this ATLAS.

The **TokenOkay** exception shall be raised in the event that the token is understood and is deliverable to the target. In other words, the token did not need to be translated by this ATLAS.

## 2.6 The Target ATLAS Interoperability Profile

The target shall indicate the specific ATLAS from which the CSS gets authorization tokens to deliver to the TSS. Once a CSS gets this profile, it shall locate the target's ATLAS. Locating an ATLAS is defined as retrieving an object reference to an **AuthTokenDispenser** interface.

The **ATLASProfile** has the following definition:

```
struct ATLASProfile {
    ATLASCacheId        the_cache_id;
    ATLASLocator        the_locator;
);
```

The field, **the_cache_id**, is a byte sequence on which the client may cache authorization tokens associated with the located ATLAS. The field, **the_locator**, shall contain a locator that leads to the object reference of the **AuthTokenDispenser** interface.

The **ATLASCacheId** is defined as a byte sequence. The caching identifier is said to be present if it is a non-empty byte sequence. The caching identifier is said not to be present if it is an empty byte sequence.

If the caching identifier is present, and the CSS caches authorization tokens, the CSS shall use the caching identifier and shall ignore the locator for the caching of authorization tokens. The locator shall not enter into the CSS caching scheme because the locator is insufficient to determine whether two ATLAS's are the same. For, example, there may be many different servers for one ATLAS, which results in many different object references and locator specifications. The caching identifier matching algorithm is byte sequence equality.

If the caching identifier is not present, the target considers the locator sufficient for caching purposes. In this case, the default matching algorithm used by the CSS is byte sequence equality on the locator. The CSS can use better matching algorithms based on its understanding of the locator and its resolution.

The target shall make the caching identity unique enough to facilitate correct client caching of authorization tokens amongst its clients. One approach would be to create a Universal Unique Identifier (UUID) [4]. For multiple targets that use the same ATLAS, it is advisable to allocate a common identifier for that ATLAS.

Targets using different privilege scopes shall have different locators, and if caching identities are supplied, they shall be different as well.

Specification of caching identifiers and procedures for allocation of caching identifiers is outside the scope of this specification.

## 2.7 Locating the Target's ATLAS

The **ATLASLocator** shall be, or shall lead to, the object reference of an ATLAS **AuthTokenDispenser** interface. Using an object reference that directly points to the ATLAS may not be desirable in all cases. In some cases, a level of indirection, such as using the CORBA Naming Service, may be useful. The **ATLASLocator** combines all these methods by using a discriminated union.

```
struct CosNamingLocator {
    CosNaming::NamingContext  name_service;
    CosNaming::Name           the_name;
};

typedef CosNaming::NamingContextExt::URLString URLocator;

typedef unsigned long ATLASLocatorType;

const  ATLASLocatorType  ATLASCosNaming  =  1;
const  ATLASLocatorType  ATLASURL        =  2;
const  ATLASLocatorType  ATLASObject     =  3;
union ATLASLocator switch (ATLASLocatorType) {
    case ATLASCosNaming:  CosNamingLocator  naming_locator;
    case ATLASURL:        URLocator         the_url;
    case ATLASObject:     AuthTokenDispenser the_dispenser;
};
```

The **ATLASCosNaming** branch of the union is a CORBA Naming Service specification in which the object reference of the naming context is supplied along with the name of the **AuthTokenDispenser**. The object reference that is resolved at the end of this name path shall resolve to a target of the **AuthTokenDispenser** type.

The **ATLASURL** branch indicates a Universal Resource Locator (URL), which is specified by the Extended Interoperable Naming Service Specification [2].

The **ATLASObject** branch indicates that an object reference points to a target of the AuthTokenDispenser type directly.

Given an **ATLASLocator** that is a URL, the client shall locate the ATLAS by resolving successive locates until it gets an object reference, because the content of the URL may be another URL. This procedure shall be followed until the URL resolution results in an invalid URL or an object reference. The object reference that results shall be that of an **AuthTokenDispenser**.

---

**Warning –** To alleviate a denial of service attack on the client directly, a client would place a limit on the number of URLs it will resolve in a chain of resolutions as well as check for loops.

---

## 2.8 The Target ATLAS Interoperability Specification

The ATLAS Interoperability Specification is contained in the **privilege_authorities** field of the **CSIIOP::SAS_ContextSec** structure. Its type is that of **CSIIOP::ServiceConfiguration**, and its definition is listed below:

```
typedef short ServiceConfigurationSyntax;

typedef sequence<octet>  ServiceSpecificName;

struct ServiceConfiguration {
    ServiceConfigurationSyntax      syntax;
    ServiceSpecificName             name;
};
```

The **syntax** field of the structure stipulates the encoding of the **name** field.

The **ServiceConfiguration** for an ATLAS is as follows:

The “syntax” field of the **ServiceConfiguration** structure shall contain the value of the following constant.

```
const CSIIOP::ServiceConfigurationSyntax SCS_ATLAS = 3;
```

The **name** field of the **ServiceConfiguration** structure shall contain the CDR encapsulation of the **ATLAS::ATLASProfile** structure.

## 2.9 Security Concerns

This section describes the security concerns one should have in both implementing, deploying, configuring, and using the ATLAS.

The ATLAS is intended to implemented with CORBA Security. Its implementations shall be security aware to support some of the operations. One such operation is the **AuthTokenDispenser::get_my_authorization_token** operation. This operation shall determine its client’s identity to return the correct authorization token for the client.

The ATLAS is a potentially sensitive service. All of its operations shall be protected by CORBA Security services and have an access control policy based on its clients’ identities. The reason the ATLAS is a sensitive service is discussed in the following subsections.

### 2.9.1 Confidentiality and Privacy

Authorization information may be a privacy concern to some individuals and organizations. For example, a nemesis discovers an individual to have privileges that allow that individual access to sensitive data. That discovery may warrant an attack on that individual to gain access to the sensitive data. Therefore, the ATLAS shall take

care and perform access control when dispensing authorization tokens. Also, where privilege information is a privacy concern in suspicious networks, the ATLAS should mandate the use of confidential security services.

The use of the ATLAS must also be considered for privacy concerns. A rogue target may know the ATLAS locator or its caching identifier. It can then spoof the CSS into giving up a cached authorization token. The CSS shall trust the target before sending authorization tokens. A rogue target may also collect authorization tokens by specifying an ATLAS for the purpose of getting the CSS to perform token translation. Therefore, the CSS shall trust the target and its located ATLAS before sending authorization tokens to the ATLAS for translation.

### 2.9.2 Integrity

To stop spoofing of targets by clients, the TSS shall verify that the authorization tokens delivered by the client are from the target's ATLAS. Also, the TSS shall make some trust determination that the tokens are valid for the particular client that delivered them.

### 2.9.3 Availability

Beyond the normal problems of dealing with denial of service attacks, there is one important concern that a CSS must take into account when trying to locate a target's ATLAS. A rogue target may put a URL loop or an exceedingly long URL resolution chain in its ATLAS locator. A CSS that is not careful may spin indefinitely trying to locate the ATLAS; and therefore, it may not do anything else. A CSS should impose a limit on chasing chains of URLs, look for loops, and possibly make a trust determination on URLs.

Another great concern for the CSS is a recursive need for authorization tokens to access an ATLAS. In the reference for interoperability the ATLAS is effectively just another target, which is protected by CORBA security services. AnATLAS's TSS may require privileges from another privilege scope. ATLAS's that require authorization tokens shall not specify themselves, or one from some mutually recursive set, as the ATLAS.

## 2.10 IllegalTokenRequest Error Codes

This section describes the error codes returned in the **IllegalTokenRequest** code that are returned from implementations. The exception has the following format:

```
exception IllegalTokenRequest {
    unsigned long     the_errnum;
    string            the_reason;
};
```

The field, **the_errnum**, contains an error code, of which the values are defined in Table 2-1. An implementation shall use the standard error codes below where possible, and indicate minor errors with the reason field. For error codes that are not standard, the code should be placed in the least significant 16 bits of the unsigned long with the following restrictions.

An error code between 300 and 400 indicates a server error, of which 300 as the default for server errors. The range 200 to 300 hundred is for errors pertaining to the client's invocation on the ATLAS. Codes between 100 and 200 are for other errors. Specific vendors may use their VMCIDs to indicate their own errors.

*Table 2-1* IllegalTokenRequest Error Codes

| Error Code | Meaning |
|---|---|
| 0100 | Generic error |
| 0200 | Generic client error |
| 0201 | Authorization token is malformed |
| 0202 | Identity token is malformed |
| 0300 | Generic server error |
| 0301 | Authorization token is not granted |
| 0302 | Authorization token type is not supported for translation |
| 0303 | Authorization token translation failed. |

# *References* *A*


[1] The Object Management Group, *Common Secure Interoperability Version 2 Joint Revised Submission*, http://cgi.omg.org/cgi-bin/doc?orbos/00-08-04, 2000

[2] The Object Management Group, *CORBA Name Service v1.1*, http://cgi.omg.org/cgi-bin/doc?orbos/99-10-11.pdf, 1999

[3] The Object Management Group, *CORBA Time Service*, http://cgi.omg.org/cgi-bin/doc?formal/97-02-22, 1997

[4] The Open Group, *Universal Unique Identifier*, http://wwww.opengroup.org/onlinepubs/9629399/adxa.htm, 1997

[5] Yergeau F., *UTF-8, a transformational format of Unicode and ISO 10646*, RFC 2044, Alis Technologies, October 1996

# A

# *Conformance Points* *B*

## *B.1 Conformance of ATLAS Implementations*

This Appendix describes the terms of conformance for implementations ofATLAS. All implementations shall implement all operations of the **AuthTokenDispenser** interface.

Implementations may not support the notion of translating authorization tokens. However, they shall still raise an IllegalTokenRequest exception with the appropriate error code defined in Section 2.10, “IllegalTokenRequest Error Codes,” on page 2-11 for the “**translate_authorization_token**” operation of the **AuthTokenDispenser** interface.

## *B.2 Conformance of Standard CORBA Security Implementations*

CORBA Security implementations that support pushing authorization tokens shall support CSS functionality defined in Section 2.3, “Reference Model for CSIv2 Authorization Interoperability,” on page 2-3” when the ATLAS profile is contained as a privilege authority within the CSS selected security mechanism component in the target IOR. CORBA Security implementations that support acceptance of authorization tokens and indicate their privilege authorities in security mechanism components of their IOR shall have implementation support to indicate an ATLAS profile as a privilege authority.

# *OMG IDL* *C*

```
// File: ATLAS.idl

#ifndef _ATLAS_IDL_
#define _ATLAS_IDL_

#include <TimeBase.idl>
#include <CosNaming.idl>
#include <CSI.idl>
#include <CSIIOP.idl>

#pragma prefix "omg.org"

module ATLAS {

typedef sequence<TimeBase::UtcT,1> ExpiryTime;

typedef sequence<CSI::IdentityToken,1> IdTokenOption;

struct AuthTokenData {
    IdTokenOption               ident_token;
    CSI::AuthorizationToken     auth_token;
    ExpiryTime                  expiry_time;
};

exception IllegalTokenRequest {
    unsigned long       the_errnum;
    string              the_reason;
};

exception TokenOkay {};

interface AuthTokenDispenser {
```

```
AuthTokenDataget_my_authorization_token()
        raises (
          IllegalTokenRequest
      );

AuthTokenDatatranslate_authorization_token(
          in   CSI::IdentityToken          the_subject,
          in   CSI::AuthorizationToken     the_token
      ) raises (
          IllegalTokenRequest,
          TokenOkay
      );

};

struct CosNamingLocator {
      CosNaming::NamingContext   name_service;
      CosNaming::Name            the_name;
};

//
// This type specifies a string encoded in UTF-8 form [IETF RFC 2044].
//
typedef sequence<octet>  UTF8String;

typedef CosNaming::NamingContextExt::URLString URLocator;

typedef unsigned long          ATLASLocatorType;

constATLASLocatorTypeATLASCosNaming=1;
const   ATLASLocatorType     ATLASURL            =   2;
const   ATLASLocatorType     ATLASObject         =   3;

union ATLASLocator switch (ATLASLocatorType) {
      case ATLASCosNaming:   CosNamingLocator    naming_locator;
      case ATLASURL:         URLocator           the_url;
      case ATLASObject:      AuthTokenDispenser  the_depenser;
};

typedef sequence<octet>        ATLASCacheId;

struct ATLASProfile {
      ATLASLocator          the_locator;
      ATLASCacheId          the_cache_id;
};

const CSIIOP::ServiceConfigurationSyntax SCS_ATLAS = 3;

};

#endif // _ATLAS_IDL_
```

# Index

# *Index*

# Authorization Token Layer Acquisition Service, v1.0 Reference Sheet

OMG documents used to create this version of the ATLAS specification:

- Submission document: orbos/01-03-07
- Final Adopted Specification: ptc/01-10-22
- FTF Report: security/02-06-01