OPC Unified Architecture

Specification

Part 2: Security Model

Release 1.01

February 6, 2009

Specification Type:	Industry Standard Specification	Comments:	Report or view errata: http://www.opcfoundation.org/errata
Title:	OPC Unified Architecture Part 2 :Security Model	Date:	February 6, 2009
Version:	Release 1.01	Software:	MS-Word
		Source:	OPC UA Part 2 - Security Model 1.01 Specification.doc
Author:	OPC Foundation	Status:	Release

CONTENTS

Page

FIGURES

TABLES

OPC FOUNDATION

UNIFIED ARCHITECTURE –

FOREWORD

This specification is the specification for developers of OPC UA applications. The specification is a result of an analysis and design process to develop a standard interface to facilitate the development of applications by multiple vendors that shall inter-operate seamlessly together.

AGREEMENT OF USE

OPC Unified Architecture Specification

Part 2: Security Model

1 Scope

This specification describes the OPC Unified Architecture (OPC UA) security model. It describes the security threats of the physical, hardware, and software environments in which OPC UA is expected to run. It describes how OPC UA relies upon other standards for security. It gives an overview of the security features that are specified in other parts of the OPC UA specification. It references services, mappings, and profiles that are specified normatively in other parts of this multi-part specification. This part of the specification is informative rather than normative. Any seeming ambiguity between this part and one of the normative parts does not remove or reduce the requirement specified in the normative part.

Note that there are many different aspects of security that have to be addressed when developing applications. However since OPC UA specifies a communication protocol, the focus is on securing the data exchanged between applications. This does not mean that an application developer can ignore the other aspects of security like protecting persistent data against tampering. It is important that the developer look into all aspects of security and decide how they can be addressed in the application.

This Part 2 is directed to readers who will develop OPC UA client or server applications or implement the OPC UA services layer.

It is assumed that the reader is familiar with Web Services and XML/SOAP. Information on these technologies can be found in SOAP Part 1: and SOAP Part 2.

2 Reference documents

2.1 References

Part 1: OPC UA Specification: Part 1 – Concepts, Version 1.01 or later

http://www.opcfoundation.org/UA/Part1/

Part 3: OPC UA Specification: Part 3 – Address Space Model, Version 1.01 or later

http://www.opcfoundation.org/UA/Part3/

Part 4: OPC UA Specification: Part 4 – Services, Version 1.01 or later

http://www.opcfoundation.org/UA/Part4/

Part 5:: OPC UA Specification: Part 5 – Information Model, Version 1.01 or later

http://www.opcfoundation.org/UA/Part5/

Part 6: OPC UA Specification: Part 6 – Mappings, Version 1.0 or later

http://www.opcfoundation.org/UA/Part6/

Part 7: OPC UA Specification: Part 7 – Profiles, Version 1.0 or later

http://www.opcfoundation.org/UA/Part7/

Part 12: OPC UA Specification: Part 12 – Discovery, Version 1.01 or later

http://www.opcfoundation.org/UA/Part12/

SOAP Part 1: SOAP Version 1.2 Part 1: Messaging Framework

http://www.w3.org/TR/soap12-part1/

SOAP Part 2: SOAP Version 1.2 Part 2: Adjuncts

http://www.w3.org/TR/soap12-part2/

XML Encryption: XML Encryption Syntax and Processing

http://www.w3.org/TR/xmlenc-core/

XML Signature:: XML-Signature Syntax and Processing

http://www.w3.org/TR/xmldsig-core/

WS Security: SOAP Message Security 1.1

http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

WS Addressing: Web Services Addressing (WS-Addressing)

http://www.w3.org/Submission/ws-addressing/

WS Trust: Web Services Trust Language (WS-Trust)

http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf

WS Secure Conversation: WebServices Secure Conversation Language (WS-SecureConversation)

http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf

SSL/TLS: RFC 2246: The TLS Protocol Version 1.0

http://www.ietf.org/rfc/rfc2246.txt

:X509: X.509 Public Key Certificate Infrastructure

http://www.itu.int/rec/T-REC-X.509-200003-I/e

HTTP: RFC 2616: Hypertext Transfer Protocol - HTTP/1.1

http://www.ietf.org/rfc/rfc2616.txt

HTTPS: RFC 2818: HTTP Over TLS

http://www.ietf.org/rfc/rfc2818.txt

IS Glossary: Internet Security Glossary

http://www.ietf.org/rfc/rfc2828.txt

NIST 800-12: Introduction to Computer Security

http://csrc.nist.gov/publications/nistpubs/800-12/

NERC CIP: CIP 002-1 through CIP 009-1, by North-American Electric Reliability Council

http://www.nerc.com/~filez/standards/Cyber-Security-Permanent.html

IEC 62351: Data and Communications Security

http://www.iec.ch/heb/d_mdoc-e050507.htm

SPP-ICS: System Protection Profile – Industrial Control System, by Process Control Security Requirements Forum (PCSRF)

http://www.isd.mel.nist.gov/projects/processcontrol/SPP-ICSv1.0.pdf

SHA-1: Secure Hash Algorithm RFC

http://tools.ietf.org/html/rfc3174

PKI: Public Key Infrastructure article in Wikipedia

http://en.wikipedia.org/wiki/Public_key_infrastructure

X509 PKI: Internet X.509 Public Key Infrastructure

http://www.ietf.org/rfc/rfc3280.txt

3 Terms, definitions, and abbreviations

3.1 OPC UA Part 1 terms

The following terms defined in Part 1 apply.

1) certificate
2) event
3) message
4) notification
5) profile
6) subscription

3.2 OPC UA Security terms

3.2.1 Application Instance

an individual installation of a program running on one computer.

NOTE: There can be several *Application Instances* of the same application running at the same time on several computers or possibly the same computer.

3.2.2 Application Instance Certificate

a *Digital Certificate* of an individual *Application Instance* that has been installed in an individual host.

NOTE: Different installations of one software product would have different *Application Instance Certificates*.

3.2.3 Asymmetric Cryptography

a *Cryptography* method that uses a pair of keys, one that is designated the *Private Key* and kept secret, the other called the *Public Key* that is generally made available.

NOTE: *Asymmetric Cryptography*, also known as "public-key cryptography". In an asymmetric encryption algorithm when an entity A wants to ensure *Confidentiality* for data it sends to another entity B, entity A encrypts the data with a *Public Key* provided by entity B. Only entity B has the matching *Private Key* that is needed to decrypt the data. In an asymmetric digital signature algorithm when an entity A wants to ensure *Integrity* or provide *Authentication* for data it sends to an entity B, entity A uses its *Private Key* to sign the data. To verify the signature, entity B uses the matching *Public Key* that entity A has provided. In an asymmetric key agreement algorithm, entity A and entity B each send their own *Public Key* to the other entity. Then each uses their own *Private Key* and the other's *Public Key* to compute the new key value. IS Glossary

3.2.4 Asymmetric Encryption

the mechanism used by *Asymmetric Cryptography* for encrypting data with the *Public Key* of an entity and for decrypting data with the associated *Private Key*.

NOTE: See Section 3.2.3 for details.

3.2.5 Asymmetric Signature

the mechanism used by *Asymmetric Cryptography* for signing data with the *Private Key* of an entity and for verifying the data's signature with the associated *Public Key*.

NOTE: See Section 3.2.3 for details.

3.2.6 Auditability

a security objective that assures that any actions or activities in a system can is recorded.

3.2.7 Auditing

the tracking of actions and activities in the system, including security related activities where the Audit records can be used to verify the operation of system security.

3.2.8 Authentication

the process of verifying the identity of an entity such as a client, server, or user.

3.2.9 Authorization

the process of granting the right or the permission to a system entity to access a system resource.

3.2.10 Availability

the running of the system with unimpeded capacity.

3.2.11 Confidentiality

the protection of data from being read by unintended parties.

3.2.12 Cryptogrophy

transforming clear, meaningful information into an enciphered, unintelligible form using an algorithm and a key.

3.2.13 Cyber Security Management System (CSMS)

a program designed by an organization to maintain the security of the entire organization's assets to an established level of *Confidentiality*, *Integrity*, and *Availability*, whether they are on the business side or the industrial automation and control systems side of the organization

3.2.14 Digital Certificate

a structure that associates an identity with an entity such as a user, a product or an *Application Instance* where the certificate has an associated asymmetric key pair which can be used to authenticate that the entity does, indeed, possess the *Private Key*.

3.2.15 Digital Signature

a value computed with a cryptographic algorithm and appended to data in such a way that any recipient of the data can use the signature to verify the data's origin and integrity.

3.2.16 Hash Function

an algorithm such as SHA-1 for which it is computationally infeasible to find either a data object that maps to a given hash result (the "one-way" property) or two data objects that map to the same hash result (the "collision-free" property). IS Glossary

3.2.17 Hashed Message Authentication Code (HMAC)

a *MAC* that has been generated using an iterative *Hash Function*.

3.2.18 Integrity

a security goal that assures that information has not been modfied or destroyed in a unauthorized manner.

NOTE: definition from IS Glossary.

3.2.19 Key Exchange Algorithm

a protocol used for establishing a secure communication path between two entities in an unsecured environment whereby both entities apply a specific algorithm to securely exchange secret keys that are used for securing the communication between them.

NOTE: A typical example of a *Key Exchange Algorithm* is the SSL Handshake Protocol specified in SSL/TLS.

3.2.20 Message Authentication Code (MAC)

a short piece of data that results from an algorithm that uses a secret key (see *Symmetric Cryptography*) to hash a message whereby the receiver of the message can check against alteration of the message by computing a *MAC* that should be identical using the same message and secret key.

3.2.21 Message Signature

a *Digital Signature* used to ensure the *Integrity* of messages that are sent between two entities.

NOTE: There are several ways to generate and verify *Message Signature*s however they can be categorized as symmetric (See Section 3.2.32) and asymmetric (See Section 3.2.5) approaches.

3.2.22 Non-Repudiation

strong and substantial evidence of the identity of the signer of a message and of message integrity, sufficient to prevent a party from successfully denying the original submission or delivery of the message and the integrity of its contents.

3.2.23 Nonce

a random number that is used once, typically by algorithms that generate security keys.

3.2.24 OPC UA Application

an OPC UA *Client*, which calls OPC UA services, or an OPC UA *Server*, which performs those services.

3.2.25 Private Key

the secret component of a pair of cryptographic keys used for *Asymmetric Cryptography*.

3.2.26 Public Key

the publicly-disclosed component of a pair of cryptographic keys used for *Asymmetric Cryptography*. IS Glossary

3.2.27 Public Key Infrastructure (PKI)

the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke *Digital Certificates* based on *Asymmetric Cryptography*.

NOTE: The core *PKI* functions are to register users and issue their public-key certificates, to revoke certificates when required, and to archive data needed to validate certificates at a much later time. Key pairs for data *Confidentiality* may be generated by a certificate authority (CA), but requiring a *Private Key* owner to generate its own key pair improves security because the *Private Key* would never be transmitted. IS Glossary. See PKI and X509 PKI for more details on *Public Key Infrastructure*s.

3.2.28 Rivest-Shamir-Adleman (RSA)

an algorithm for *Asymmetric Cryptography*, invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. IS Glossary

3.2.29 Secure Channel

in OPC UA, a communication path established between an OPC UA client and server that have authenticated each other using certain OPC UA services and for which security parameters have been negotiated and applied.

3.2.30 Symmetric Cryptography

A branch of cryptography involving algorithms that use the same key for two different steps of the algorithm (such as encryption and decryption, or Signature creation and signature verification). IS Glossary

3.2.31 Symmetric Encryption

the mechanism used by *Symmetric Cryptography* for encrypting and decrypting data with a cryptographic key shared by two entities.

3.2.32 Symmetric Signature

the mechanism used by *Symmetric Cryptography* for signing data with a *cryptographic key* shared by two entities.

NOTE: The signature is then validated by generating the signature for the data again and comparing these two signatures. If they are the same then the signature is valid, otherwise either the key or the data is different from the two entities. Section 3.2.17 defines a typical example for an algorithm that generates *Symmetric Signature*s.

3.2.33 X.509 Certificate

a *Digital Certificate* in one of the formats defined by X.509 v1, 2, or 3. An *X.509 Certificate* contains a sequence of data items and has a digital signature computed on that sequence.

3.3 Abbreviations and symbols

CSMS	Cyber Security Management System
DSA	Digital Signature Algorithm
PKI	Public Key Infrastructure
RSA	public key algorithm for signing or encryption, Rivest, Shamir, Adleman
SHA1	Secure Hash Algorithm 1
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UA	Unified Architecture
URI	Uniform Resource Identifier
XML	Extensible Mark-up Language

3.4 Conventions

3.4.1 Conventions for security model figures

The figures in this document do not use any special common conventions. Any conventions used in a particular figure are explained for that figure.

4 OPC UA Security architecture

4.1 OPC UA Security Environment

OPC UA is a protocol used between components in the operation of an industrial facility at multiple levels: from high-level enterprise management to low-level direct process control of a device. The use of OPC UA for enterprise management involves dealings with customers and suppliers. It may be an attractive target for industrial espionage or sabotage and may also be exposed to threats through untargeted malware, such as worms, circulating on public networks. Disruption of communications at the process control end causes at least an economic cost to the enterprise and can have employee and public safety consequences or cause environmental damage. This may be an attractive target for those who seek to harm the enterprise or society.

OPC UA will be deployed in a diverse range of operational environments, with varying assumptions about threats and accessibility, and with a variety of security policies and enforcement regimes. OPC UA, therefore, provides a flexible set of security mechanisms. Figure 1 is a composite that shows a combination of such environments. Some OPC UA clients and servers are on the same host and can be more easily protected from external attack. Some clients and servers are on different hosts in the same operations network and might be protected by the security boundary protections that separate the operations network from external connections. Some *OPC UA Applications* run in relatively open environments where users and applications might be difficult to control. Other applications are embedded in control systems that have no direct electronic connection to external systems.

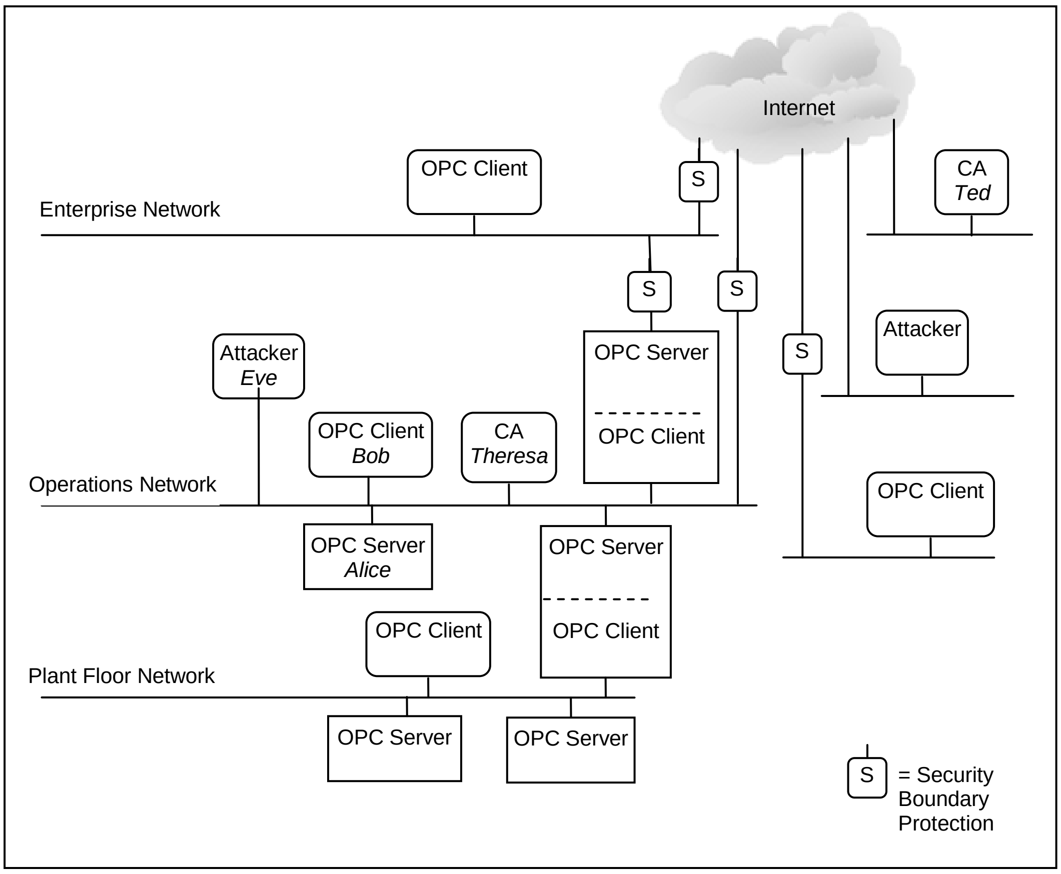


Figure 1 - OPC UA Network Model

*OPC UA Application*s may run at different places in this wide range of environments. Client and server may communicate within the same host or between hosts within the highly protected control network. Alternatively, OPC UA clients and servers may communicate in the much more open environment over the Internet. The OPC UA activities are protected by whatever security controls the site provides to the parts of the system within which *OPC UA Application*s run.

4.2 Security Objectives

4.2.1 Overview

Fundamentally, information system security reduces the risk of damage from attacks. It does this by identifying the threats to the system, identifying the system’s vulnerabilities to these threats, and providing countermeasures. The countermeasures reduce vulnerabilities directly, counteract threats, or recover from successful attacks.

Industrial automation system security is achieved by meeting a set of objectives. These objectives have been refined through many years of experience in providing security for information systems in general and they remain quite constant despite the ever-changing set of threats to systems. They are described in the following subsections of Section 4.2. Following the sections that describe the OPC UA security architecture and functions, Section 5.2 reconciles these objectives against the OPC UA functions.

4.2.2 Authentication

Entities such as clients, servers, and users should prove their identities. *Authentication* can be based on something the entity is, has, or knows.

4.2.3 Authorization

The access to read, write, or execute resources should be authorized for only those entities that have a need for that access within the requirements of the system. *Authorization* can be as coarse-grained as allowing or disallowing a client to call a server or it could be much finer grained, such as allowing specific actions on specific information items by specific users.

4.2.4 Confidentiality

Data must be protected from passive attacks, such as eavesdropping, whether the data is being transmitted, in memory, or being stored. To provide *Confidentiality* data encryption algorithms using special secrets for securing data are used together with authentication and authorization mechanisms for accessing that secret.

4.2.5 Integrity

Receivers must receive the same information that the sender sent, without the data being changed during transmission.

4.2.6 Auditability

Actions taken by a system have to be recorded in order to provide evidence to stakeholders that this system works as intended and to identify the initiator of certain actions..

4.2.7 Availability

Availability is impaired when the execution of software that needs to run is turned off or when software or the communication system is overwhelmed processing input. Impaired *Availability* in OPC UA can appear as slowing down of subscription performance or inability to add sessions for example.

4.3 Security Threats to OPC UA Systems

4.3.1 Overview

OPC UA provides countermeasures to resist the threats to the security of the information that is communicated. The following subsections of Section 4.3 list the currently known threats to environments in which OPC UA will be deployed. Following the sections that describe the OPC UA security architecture and functions, Section 5.1 reconciles these threats against the OPC UA functions.

4.3.2 Message Flooding

An attacker can send a large volume of messages, or a single message that contains a large number of requests, with the goal of overwhelming the OPC UA server or components on which the OPC UA server may depend for reliable operation such as CPU, TCP/IP stack, Operating System, or the File System. Flooding attacks can be conducted at multiple layers including OPC UA, SOAP, [HTTP] or TCP.

Message flooding attacks can use both well-formed and malformed messages. In the first scenario the attacker could be a malicious person using a legitimate client to flood the server with requests. Two cases exist, one in which the client does not have a session with the server and one in which it does. Message flooding may impair the ability to establish OPC UA sessions, or terminate an existing session. In the second scenario an attacker could use a malicious client that floods an OPC UA server with malformed messages in order to exhaust the server’s resources.

More generally message flooding may impair the ability to communicate with an OPC UA entity and result in denial of service.

Message flooding impacts *Availability*.

See clause 5.1.2 for the reconciliation of this threat.

4.3.3 Eavesdropping

Eavesdropping is the unauthorized disclosure of sensitive information that might result directly in a critical security breach or be used in follow-on attacks.

If an attacker has compromised the underlying operating system or the network infrastructure, the attacker might record and capture messages. It may be beyond the capability of a client or server to recover from a compromise of the operating system.

Eavesdropping impacts *Confidentiality* directly and threatens all of the other security objectives indirectly.

See clause 5.1.3 for the reconciliation of this threat.

4.3.4 Message Spoofing

An attacker may forge messages from a client or a server. Spoofing may occur at multiple layers in the protocol stack.

By spoofing messages from a client or a server, attackers may perform unauthorized operations and avoid detection of their activities.

Message spoofing impacts *Integrity* and *Authorization*.

See clause 5.1.4 for the reconciliation of this threat.

4.3.5 Message Alteration

Network traffic and application layer messages may be captured, modified, and the modified message sent forward to OPC UA clients and servers. Message alteration may allow illegitimate access to a system.

Message alteration impacts *Integrity* and *Authorization*.

See clause 5.1.5 for the reconciliation of this threat.

4.3.6 Message Replay

Network traffic and valid application layer messages may be captured and resent to OPC UA clients and servers at a later stage without modification. An attacker could misinform the user or send in improper command such a command to open a valve but at an improper time.

Message replay impacts *Authorization*.

See clause 5.1.6 for the reconciliation of this threat.

4.3.7 Malformed Messages

An attacker can craft a variety of messages with invalid message structure (malformed XML, SOAP, UA Binary, etc.) or data values and send them to OPC UA clients or servers.

The OPC UA client or server may incorrectly handle certain malformed messages by performing unauthorized operations or processing unnecessary information. It might result in a denial or degradation of service including termination of the application or, in the case of embedded devices, a complete crash. In a worst case scenario an attacker could also use malformed messages as a pre-step for a multi-level attack to gain access to the underlying system of an OPC UA application.

Malformed messages impact *Integrity* and *Availability*.

See clause 5.1.7 for the reconciliation of this threat.

4.3.8 Server Profiling

An attacker tries to deduce the identity, type, software version, or vendor of the server or client in order to apply knowledge about specific vulnerabilities of that product to mount a more intrusive or damaging attack. The attacker might profile the target by sending valid or invalid formatted messages to the target and try to recognize the type of target by the pattern of its normal and error responses.

Server profiling impacts all of the objectives indirectly.

See clause 5.1.8 for the reconciliation of this threat.

4.3.9 Session Hijacking

An attacker may use information (retrieved by sniffing the communication or by guessing) about a running session established between two applications to inject manipulated messages (with valid session information) that allow him to take over the session from the authorized user.

An attacker may gain unauthorized access to data or perform unauthorized operations.

Session hijacking impacts all of the objectives.

See clause 5.1.9 for the reconciliation of this threat.

4.3.10 Rogue Server

An attacker builds a malicious OPC UA server or installs an unauthorized instance of a genuine OPC UA server.

The OPC client may disclose necessary information.

A rogue server impacts all of the security objectives except *Integrity*.

See clause 5.1.10 for the reconciliation of this threat.

4.3.11 Compromising User Credentials

An attacker obtains user credentials such as usernames, passwords, certificates, or keys by observing them on papers, on screens, or in electronic communications; by cracking them through guessing or the use of automated tools such as password crackers.

An unauthorized user could launch and access the system to obtain all information and make control and data changes that harm plant operation or information. Once compromised credentials are used, subsequent activities may all appear legitimate.

Compromised user credentials impact *Authorization* and *Confidentiality*.

See clause 5.1.11 for the reconciliation of this threat.

4.4 OPC UA Relationship to Site Security

OPC UA security works within the overall *Cyber Security Management System* (*CSMS*) of a site. Sites often have a *CSMS* that addresses security policy and procedures, personnel, responsibilities, audits, and physical security. A *CSMS* typically addresses threats that include those that were described in Section 4.3. They also analyze the security risks and determine what security controls the site needs.

Resulting security controls commonly implement a “defence-in-depth” strategy that provides multiple layers of protection and recognizes that no single layer can protect against all attacks. Boundary protections, shown as abstract examples in Figure 1, may include firewalls, intrusion detection and prevention systems, controls on dial-in connections, and controls on media and computers that are brought into the system. Protections in components of the system may include hardened configuration of the operating systems, security patch management, anti-virus programs, and not allowing email in the control network. Standards that may be followed by a site include [NERC CIP] and [IEC 62351] which are referenced in Section 2.

The security requirements of a site *CSMS* apply to its OPC UA interfaces. That is, the security requirements of the OPC UA interfaces that are deployed at a site are specified by the site, not by the OPC UA specification. OPC UA specifies features that are intended so that conformant client and server products can meet the security requirements that are expected to be made by sites where they will be deployed. Those who are responsible for the security at the site should determine how to meet the site requirements with OPC UA conformant products.

The system owner that installs OPC UA clients or servers should analyze its security risks and provide appropriate mechanisms to mitigate those risks to achieve an acceptable level of security. OPC UA meets the wide variety of security needs that might result from such individual analyses. OPC UA clients and servers are required to be implemented with certain security features, which are available for the system owner’s optional use. Each system owner should be able to tailor a security solution that meets its security and economic requirements using a combination of mechanisms available within the OPC UA specification and external to OPC UA.

The security requirements placed on the OPC UA clients and servers deployed at a site are specified by the site *CSMS*, not by the OPC UA specification. The OPC UA security specifications, however, are requirements placed upon OPC UA client and server products, and recommendations of how OPC UA should be deployed at a site in order to meet the security requirements that are anticipated to be specified at the site.

OPC UA addresses some threats as described in the section 4.3. The OPC foundation recommends that vendors address the remaining threats, as detailed in section 6. Threats to infrastructure components that might result in the compromise of client and server operating systems are not addressed by OPC UA.

4.5 OPC UA Security Architecture

The OPC UA security architecture is a generic solution that allows implementation of the required security features at various places in the *OPC UA Application* architecture. Depending on the different mappings described in Part 6, the security objectives are addressed at different levels. The OPC UA Security Architecture is structured in an Application Layer and a Communication Layer atop the Transport Layer as shown in Figure 2.

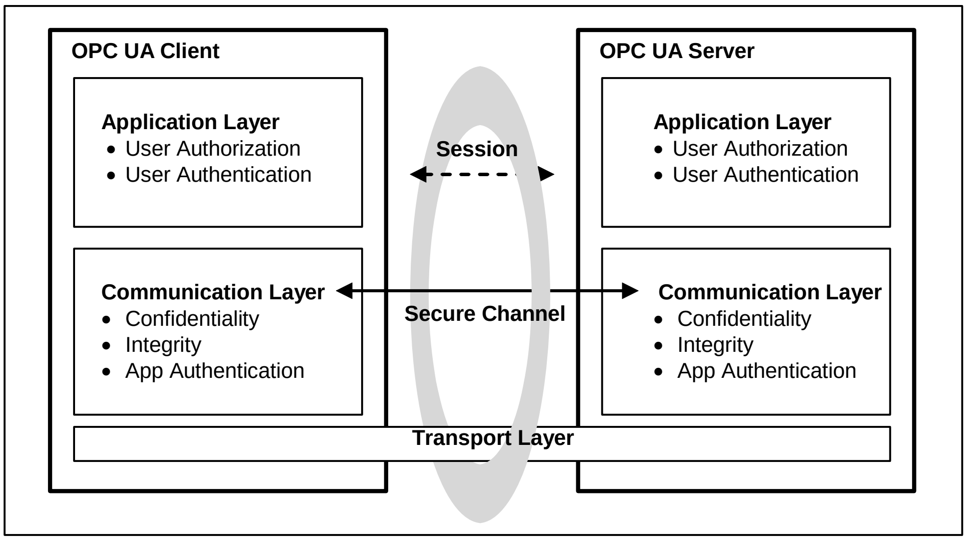


Figure 2 – OPC UA Security Architecture

The routine work of a client application and a server application to transmit plant information, settings, and commands is done in a session in the Application Layer. The Application Layer also manages the security objectives user *Authentication* and user *Authorization*. The security objectives that are managed by the Application Layer are addressed by the Session Services that are specified in Part 4. A session in the Application Layer communicates over a *Secure Channel* that is created in the Communication Layer and relies upon it for secure communication. All of the session data is passed to the Communication Layer for further processing.

Although a session communicates over a *Secure Channel and has to be activated before it can be used*, the binding of users, sessions, and *Secure Channels* is flexible.

Impersonation allows the user of the session to change. A session can have a different user than the user that activated the *session for the first time, since user credentials are not validated* before activating a session.

When a *Secure Channel* breaks, the session will still be valid to be able to re-establish the *Secure Channel* otherwise the session closes after its lifetime expires.

The Communication Layer provides security mechanisms to meet *Confidentiality*, *Integrity* and application *Authentication* as security objectives.

One essential mechanism to meet the above mentioned security objectives is to establish a *Secure Channel* (see Section 4.11) that is used to secure the communication between a client and a server. The *Secure Channel* provides encryption to maintain *Confidentiality*, *Message Signature*s to maintain *Integrity* and *Digital Certificates* to provide application *Authentication* for data that comes from the Application Layer and passes the “secured” data to the Transport Layer. The security mechanisms that are managed by the Communication Layer are provided by the *Secure Channel* Services that are specified in Part 4.

The security mechanisms provided by the *Secure Channel* services are implemented by a protocol stack that is chosen for the implementation. Mappings of the services to some of the protocol stack options are specified in Part 6 which details how the functions of the protocol stack are used to meet the OPC UA security objectives.

The Communication Layer can represent an OPC UA protocol stack. OPC UA specifies two alternative stack mappings that can be used as the Communication Layer. These mappings are OPC UA Native mapping and Web Services mapping.

If the OPC UA Native mapping is used, then functionalities for *Confidentiality*, *Integrity*, application *Authentication*, and the *Secure Channel* are similar to the SSL/TLS specifications, as described in detail in Part 6.

If the Web Services mapping is used, then WS Security, WS Secure Conversation and XML Encryption as well as XML Signature: are used to implement the mechanisms for *Confidentiality*, *Integrity*, application *Authentication* as well as for implementing a *Secure Channel*. For more specific information, see Part 6.

The Transport Layer handles the transmission, reception and the transport of data that is provided by the Communication Layer.

To survive the loss of the Transport Layer connections (e.g. TCP connections) and resume with another, the implementation of the Communication Layer is responsible to re-establish the Transport Layer connection without interrupting the logical *Secure Channel*.

4.6 Security Policies

Security policies specify which security mechanisms are to be used and are derived from Security Profiles (see section 4.7 for details). Security policies are used by the server to announce what mechanisms it supports and by the client to select one of those available policies to be used for the *Secure Channel* it wishes to open. The policies specified include the following:

- algorithms for signing and encryption
- algorithm for key derivation

The choice of policy is normally made by the control system administrator, typically when the client and server products are installed.

The announcement of security policies is handled by special discovery services specified in Part 4. More details about the discovery mechanisms and policy announcement strategies can be found in Part 12

If a server serves multiple clients, it maintains separate policy selections for the different clients. This allows a new client to select policies independent of the policy choices that other clients have selected for their *Secure Channels*.

Since computing power increases over the years specific algorithms that are considered as secure today can become insecure in the future, therefore it makes sense to support different security policies in an *OPC UA Application* to be able to migrate forward in such a case.

There is also the case that new security policies are composed to support new algorithms that improve the level of security of OPC UA products. The application architecture of OPC UA clients and servers should be designed in a way that it is possible to update or add additional cryptographic algorithms to the application.

Part 7 specifies several policies which are identified by a specific unique URI. To improve interoperability among vendors’ products, server products should implement these policies rather than define their own.

4.7 Security Profiles

OPC UA client and server products are certified against profiles that are defined in Part 7. Some of the profiles specify security functions and others specify other functionality that is not related to security. The profiles impose requirements on the certified products but they do not impose requirements on how the products are used. A consistent minimum level of security is required by the various profiles. However, different profiles specify different details, such as which encryption algorithms are required for which OPC UA functions. If a problem is found in one encryption algorithm, then the OPC foundation can define a new profile that is similar, but that specifies a different encryption algorithm that does not have a known problem. Part 7, not this Part 2, is the normative specification of the profiles.

Policies refer to many of the same security choices as profiles, however the policy specifies which of those choices to use in the session. The policy does not specify the range of choices that the product offers like the profile does.

Each security mechanism in OPC UA is provided in client and server products in accordance with the profiles with which the client or server complies. At the site, however, the security mechanisms may be deployed optionally. In this way each individual site has all of the OPC UA security functions available and can choose which of them to use to meet its security objectives.

4.8 User Authorization

OPC UA provides a mechanism to exchange user credentials but does not specify how the applications use these credentials. Client and server applications may determine in their own way what data is accessible and what operations are authorized.

4.9 User Authentication

User *Authentication* is provided by the Session Services with which the client passes user credentials to the server as specified in Part 4. The server can authenticate the user with these credentials.

The user who is communicating over a session can be changed using the ActivateSession service in order to meet needs of the application.

4.10 Application Authentication

OPC UA uses a concept conveying Application *Authentication* to allow applications that intend to communicate to identify each other. Each *OPC UA Application Instance* has a *Digital Certificate* (*Application Instance Certificate*) assigned that is exchanged during *Secure Channel* establishment. The receiver of the *Certificate* checks whether it trusts it and based on this check it accepts or rejects the request or response message from the sender.

More details on *Application Authentication* can be found in Part 4.

4.11 OPC UA Security Related Services

The OPC UA Security Services are a group of abstract service definitions specified in Part 4 that are used for applying various security mechanisms to communication between OPC UA clients and servers.

The Discovery Service Set (specified in Part 4) defines services used by an OPC UA client to inform itself about the security policies (see Section 4.6) and the *Digital Certificate*s of specific OPC UA servers.

The services of the *Secure Channel* Service Set (specified in Part 4) are used to establish a *Secure Channel* which is responsible for securing messages sent between a client and a server. The challenge of the *Secure Channel* establishment is that it requires the client and the server to securely exchange cryptographic keys and secret information in an insecure environment, therefore a specific *Key Exchange Algorithm* (similar to SSL Handshake protocol defined in SSL/TLS) is applied by the communication participants:

The OPC UA client retrieves the security policies and *Digital Certificate*s of the OPC UA server by the above mentioned discovery services. These *Digital Certificates* contain the *Public Keys* of the OPC UA server.

The OPC UA client sends its *Public Key* in a *Digital Certificate* and secret information with the OpenSecureChannel service message to the server. This message is secured by applying *Asymmetric Encryption with the server's Public Key* and by *generating Asymmetric Signatures* with the client's *Private Key.* However the *Digital Certificate* is sent unencrypted so that the receiver can use it to verify the *Asymmetric Signature*.

The server decrypts the message with its *Private Key* and verifies the *Asymmetric Signature* with the client's *Public Key*. The secret information of the OPC UA client together with the secret information of the server is used to derive a set of cryptographic keys that are used for securing all further messages. Furthermore all other service messages are secured with *Symmetric Encryption* and *Symmetric Signature*s instead of the asymmetric equivalents.

The server sends its secret information in the service response to the client so that the client can derive the same set of cryptographic keys.

Since client and server have the same set of cryptographic keys they can communicate in a secure way with each other.

These derived cryptographic keys are changed periodically so that attackers do not have unlimited time and unrestricted sequences of messages to use to determine what the keys are.

4.12 Auditing

4.12.1 General

Clients and servers generate audit records of successful and unsuccessful connection attempts, results of security option negotiations, configuration changes, system changes, and session rejections.

OPC UA provides support for security audit trails through two mechanisms. First, it provides for traceability between client and server audit logs. The client generates an audit log entry for an operation that includes a request. When the client issues a service request, it generates an audit log entry and includes the local identifier of the log entry in the request sent to the server. The server logs requests that it receives and includes the client's entry id in its audit log entry. In this fashion, if a security-related problem is detected at the server, the associated client audit log entry can be located and examined. OPC UA does not require the audit entries to be written to disk, but it does require that they be available. OPC UA provides the capability for servers to generate event notifications that report auditable events to clients capable of processing and logging them. See Part 4 for more details on how services in OPC UA are audited.

Second, OPC UA defines audit parameters to be included in audit records. This promotes consistency across audit logs and in audit events. Part 5: defines the data types for these parameters. Other information models may extend the audit definitions. Part 7 defines profiles, which include the ability to generate audit events and use these parameters, including the client audit record id.

Because the audit logs are used to prove that the system is operating securely, the audit logs themselves must also be secured from unauthorized tampering. If someone without authorization were able to alter or delete log records, this could hide an actual or attempted security breach. Because there are many different ways to generate and store audit logs (e.g. files or database), the mechanisms to secure audit logs are outside the scope of this specification.

The following clauses illustrate the behaviour of OPC UA servers and clients that support *Auditing*.

4.12.2 Single client and server

Figure 3 illustrates the simple case of a client communicating with a server.

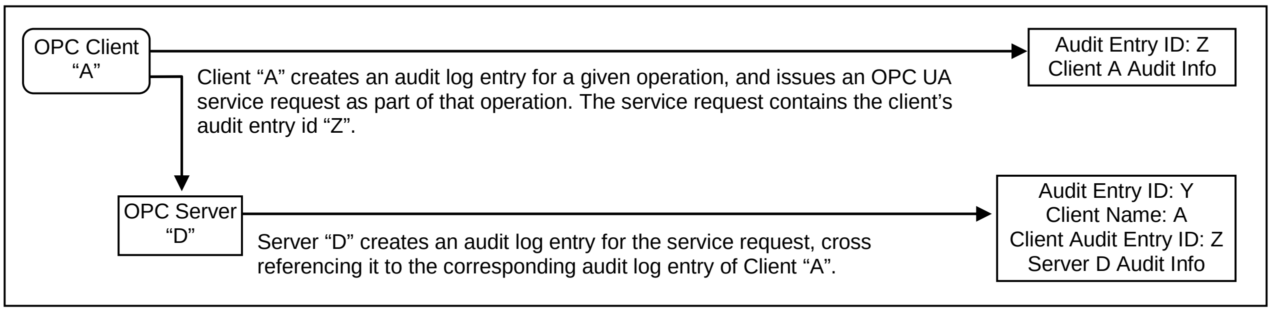

OPC Client "A"
Audit Entry ID: Z
Client A Audit Info
Client "A" creates an audit log entry for a given operation, and issues an OPC UA service request as part of that operation. The service request contains the client's audit entry id "Z".
OPC Server "D"
Audit Entry ID: Y
Client Name: A
Client Audit Entry ID: Z
Server D Audit Info
Server "D" creates an audit log entry for the service request, cross referencing it to the corresponding audit log entry of Client "A".

Figure 3 – Simple Servers

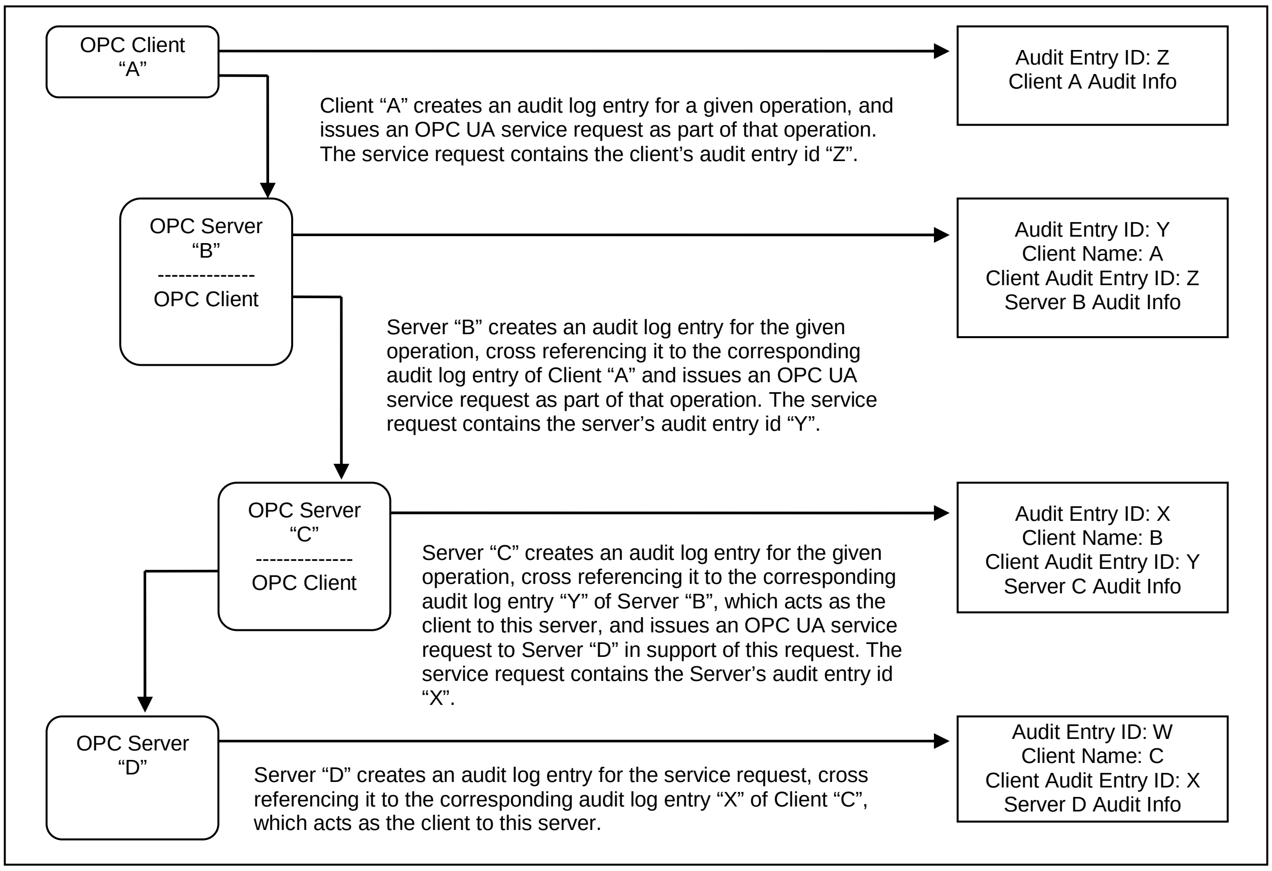
In this case, client "A" executes some auditable operation that includes the invocation of an OPC UA service in Server "D". It writes its own audit log entry, and includes the identifier of that entry in the service request that it submits to the server.

The server receives the request and creates its own audit log entry for it. This entry is identified by its own audit id and contains its own *Auditing* information. It also includes the name of the client that issued the service request and the client audit entry id received in the request.

Using this information, an auditor can inspect the collection of log entries of the server and relate them back to their associated client entries.

4.12.3 Aggregating server

Figure 4 illustrates the case of a client accessing services from an aggregating server. An aggregating server is a server that provides its services by accessing services of other OPC UA servers, referred to as lower layer-servers.


OPC Client "A"
Audit Entry ID: Z
Client A Audit Info
Client "A" creates an audit log entry for a given operation, and issues an OPC UA service request as part of that operation. The service request contains the client's audit entry id "Z".
OPC Server "B"

OPC Client
Audit Entry ID: Y
Client Name: A
Client Audit Entry ID: Z
Server B Audit Info
Server "B" creates an audit log entry for the given operation, cross referencing it to the corresponding audit log entry of Client "A" and issues an OPC UA service request as part of that operation. The service request contains the server's audit entry id "Y".
OPC Server "C"

OPC Client
Audit Entry ID: X
Client Name: B
Client Audit Entry ID: Y
Server C Audit Info
Server "C" creates an audit log entry for the given operation, cross referencing it to the corresponding audit log entry "Y" of Server "B", which acts as the client to this server, and issues an OPC UA service request to Server "D" in support of this request. The service request contains the Server's audit entry id "X".
OPC Server "D"
Audit Entry ID: W
Client Name: C
Client Audit Entry ID: X
Server D Audit Info
Server "D" creates an audit log entry for the service request, cross referencing it to the corresponding audit log entry "X" of Client "C", which acts as the client to this server.

Figure 4 – Aggregating Servers

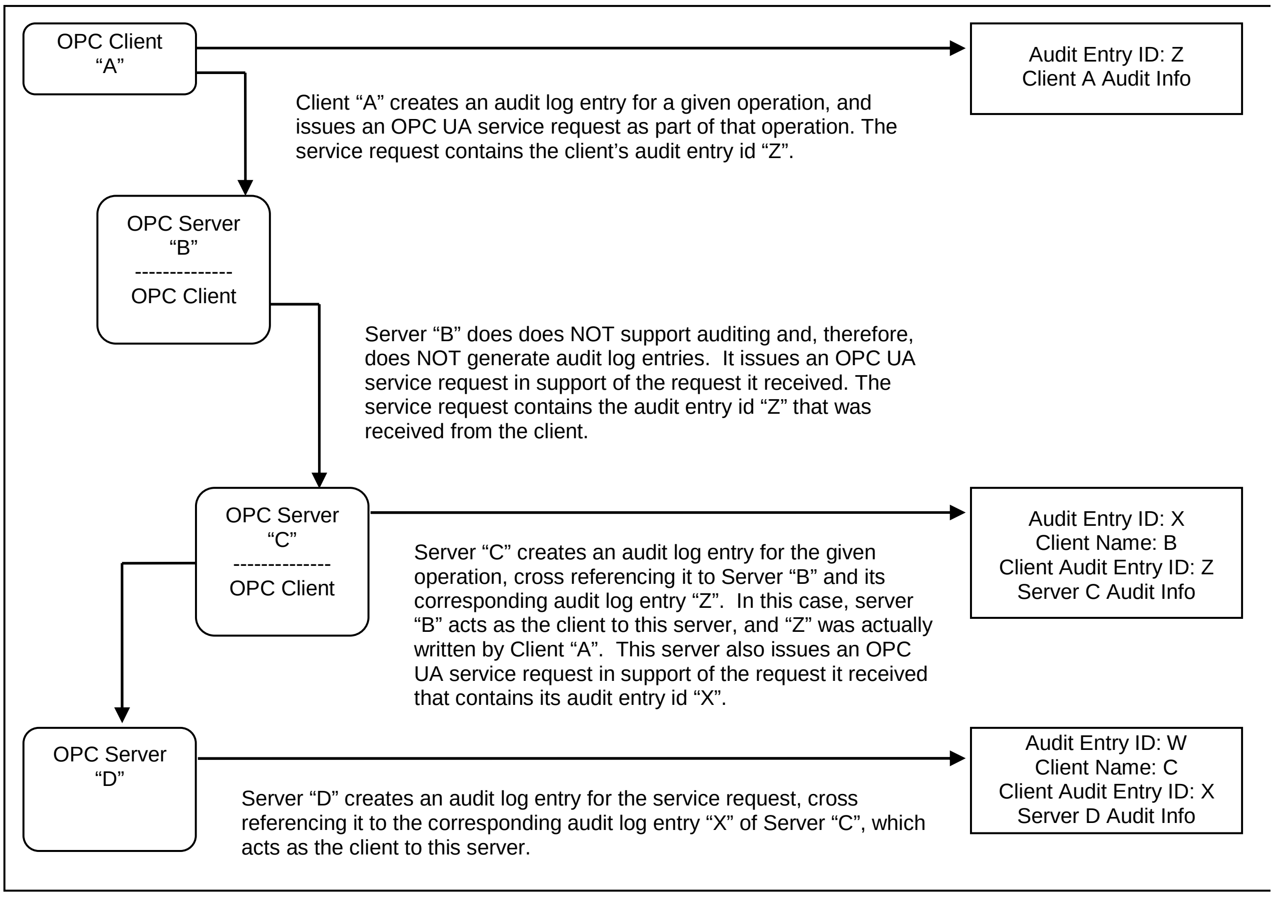
In this case, each of the servers receives requests and creates its own audit log entry for them. Each entry is identified by its own audit id and contains its own *Auditing* information. It also includes the name of the client that issued the service request and the client audit entry id received in the request. The server then passes the audit id of the entry it just created to the next server in the chain.

Using this information, an auditor can inspect the server's log entries and relate them back to their associated client entries.

In most cases the servers will only generate audit events, but these audit events will still contain the same information as the audit log records. In the case of aggregating servers a server would also be required to subscribe for audit events from the servers it is aggregating. In this manner, server "B" would be able to provide all of the audit events to client "A", including the event generated by server "C" and server "D"

4.12.4 Aggregation through a non-auditing server

Figure 5 illustrates the case of a client accessing services from an aggregating server that does not support *Auditing*.



OPC Client "A"
Audit Entry ID: Z
Client A Audit Info
Client "A" creates an audit log entry for a given operation, and issues an OPC UA service request as part of that operation. The service request contains the client's audit entry id "Z".
OPC Server "B" -------------- OPC Client
Server "B" does does NOT support auditing and, therefore, does NOT generate audit log entries. It issues an OPC UA service request in support of the request it received. The service request contains the audit entry id "Z" that was received from the client.
OPC Server "C" -------------- OPC Client
Audit Entry ID: X
Client Name: B
Client Audit Entry ID: Z
Server C Audit Info
Server "C" creates an audit log entry for the given operation, cross referencing it to Server "B" and its corresponding audit log entry "Z". In this case, server "B" acts as the client to this server, and "Z" was actually written by Client "A". This server also issues an OPC UA service request in support of the request it received that contains its audit entry id "X".
OPC Server "D"
Audit Entry ID: W
Client Name: C
Client Audit Entry ID: X
Server D Audit Info
Server "D" creates an audit log entry for the service request, cross referencing it to the corresponding audit log entry "X" of Server "C", which acts as the client to this server.

Figure 5 – Aggregation with a Non-Auditing Server

In this case, each of the servers receives their requests and creates their own audit log entry for them, with the exception of server "B", which does not support *Auditing*. In this case, server "B" passes the audit id it receives from its client "A" to the next server. This creates the required audit chain. Server "B" is not listed as supporting *Auditing*. In a case where a server does not support writing audit entries, the entire system may be considered as not supporting *Auditing*.

In the case of an aggregating server that does not support *Auditing*, the server would still be required to subscribe for audit events from the servers it is aggregating. In this manner, server "B" would be able to provide all of the audit events to client "A", including the event generated by server "C" and server "D", even though it did not generate an audit event.

4.12.5 Aggregating server with service distribution

Figure 6 illustrates the case of a client that submits a service request to an aggregating server, and the aggregating service supports that service by submitting multiple service requests to its underlying servers.

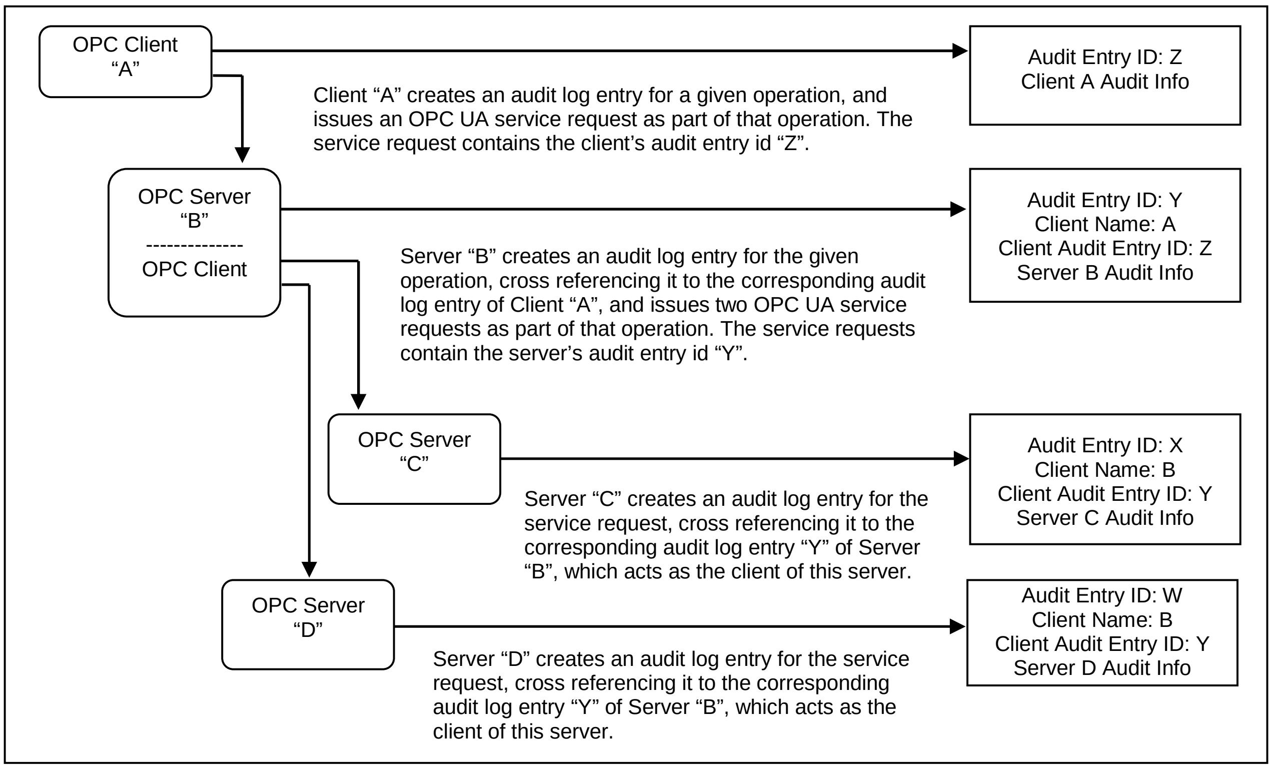

OPC Client "A"
Audit Entry ID: Z
Client A Audit Info
Client "A" creates an audit log entry for a given operation, and issues an OPC UA service request as part of that operation. The service request contains the client's audit entry id "Z".
OPC Server "B" -------------- OPC Client
Audit Entry ID: Y
Client Name: A
Client Audit Entry ID: Z
Server B Audit Info
Server "B" creates an audit log entry for the given operation, cross referencing it to the corresponding audit log entry of Client "A", and issues two OPC UA service requests as part of that operation. The service requests contain the server's audit entry id "Y".
OPC Server "C"
Audit Entry ID: X
Client Name: B
Client Audit Entry ID: Y
Server C Audit Info
Server "C" creates an audit log entry for the service request, cross referencing it to the corresponding audit log entry "Y" of Server "B", which acts as the client of this server.
OPC Server "D"
Audit Entry ID: W
Client Name: B
Client Audit Entry ID: Y
Server D Audit Info
Server "D" creates an audit log entry for the service request, cross referencing it to the corresponding audit log entry "Y" of Server "B", which acts as the client of this server.

Figure 6 – Aggregate server with Service Distribution

In the case of aggregating servers a server would also be required to subscribe for audit events from the servers it is aggregating. In this manner, server "B" would be able to provide all of the audit events to client "A", including the event generated by server "C" and server "D"

5 Security Reconciliation

5.1 Reconciliation of Threats with OPC UA Security Mechanisms

5.1.1 Overview

The following subsections of Section 5.1 reconcile the threats that were described in Section 4.3 against the OPC UA functions. Each of the following subsections relates directly to the threat described in its corresponding subsection of Section 4.3.

Compared to the reconciliation with the objectives that will be given in section 5.2, this is a more specific reconciliation that relates OPC UA security functions to specific threats.

5.1.2 Message Flooding

See clause 4.3.2 for a description of this threat.

OPC UA minimizes the loss of *Availability* caused by message flooding by minimizing the amount of processing done with a message before the message is authenticated. This prevents an attacker from leveraging a small amount of effort to cause the legitimate *OPC UA Application* to spend a large amount of time responding, thus taking away processing resources from legitimate activities.

GetEndpoints (specified in Part 4) and OpenSecureChannel (specified in Part 4) are the only services that the server handles before the client is recognized. The response to GetEndpoints is only a set of static information so the server does not need to do much processing. The response to OpenSecureChannel consumes significant server resources because of the signature and encryption processing. OPC UA has minimized this processing, but it can not be eliminated.

The server implementation could protect itself from floods of OpenSecureChannel messages in two ways.

First, the server could intentionally delay its processing of OpenSecureChannel requests once it receives a bad one and issuing an alarm would alert plant personnel that an attack is underway that could still be blocking new legitimate OpenSecureChannel calls.

Second, when an OpenSecureChannel request attempts to exceed the server’s specified maximum number of concurrent channels the server replies with an error response without performing the signature and encryption processing. Certified OPC UA servers are required to specify their maximum number of concurrent channels in their product documentation as specified in Part 7.

OPC UA user and client *Authentication* reduce the risk of a legitimate client being used to mount a flooding attack. See the reconciliation of *Authentication* in Section 5.2.2.

OPC UA *Auditing* functionality provides the site with evidence that can help the site discover that flooding attacks are being mounted and find ways to prevent similar future attacks. (See Section 4.12)

OPC UA relies upon the site *CSMS* to prevent attacks such as message flooding at protocol layers and systems that support OPC UA.

5.1.3 Eavesdropping

See clause 4.3.3 for a description of this threat.

OPC UA provides encryption to protect against eavesdropping as described in Section 5.2.4 - *Confidentiality*.

5.1.4 Message Spoofing

See clause 4.3.4 for a description of this threat.

As specified in Part 4 and Part 6, OPC UA counters message spoofing threats by the possibility to sign messages. Additionally messages will always contain a valid Session ID, *Secure Channel* ID, Request ID as well as the correct Sequence Number.

5.1.5 Message Alteration

See clause 4.3.5 for a description of this threat.

OPC UA counters message alteration by the signing of messages that are specified in Part 4. If messages are altered, checking the signature will reveal any changes and allow the recipient to discard the message.

5.1.6 Message Replay

See clause 4.3.6 for a description of this threat.

OPC UA uses Session IDs, *Secure Channel* IDs, Timestamps, Sequence Numbers and Request IDs for every request and response message. Messages are signed and cannot be changed without detection, therefore it would be very hard to replay a message, such that the message would have a valid Session ID, *Secure Channel* ID, Timestamp, Sequence Numbers and Request ID. (All of which are specified in Part 4 and Part 6)

5.1.7 Malformed Messages

See clause 4.3.7 for a description of this threat.

Implementations of OPC UA client and server products counter threats of malformed messages by checking that messages have the proper form and that parameters of messages are within their legal range. This is specified in Part 4 and Part 6.

5.1.8 Server Profiling

See clause 4.3.8 for a description of this threat.

OPC UA limits the amount of information that servers provide to clients that have not yet been identified. This information is the response to the GetEndpoints service specified in Part 4.

5.1.9 Session Hijacking

See clause 4.3.9 for a description of this threat.

OPC UA counters session hijacking by assigning a security context (i.e. *Secure Channel*) with each session as specified in the CreateSession service in Part 4. Hijacking a session would thus first require compromising the security context.

5.1.10 Rogue Server

See clause 4.3.10 for a description of this threat.

OPC UA client applications counter the use of rogue servers by validating server *Application Instance Certificates*. There would still be the possibility that a rogue server provides a certificate from a certified OPC UA server, but since it does not possess the appropriate *Private Key* (because this will never be distributed) to decrypt and verify messages secured with the correct *Public Key* the rogue server would never be able to read and misuse secured data sent by a client.

5.1.11 Compromising User Credentials

See clause 4.3.11 for a description of this threat.

OPC UA protects user credentials sent over the network by encryption as described in Section 5.2.4.

OPC UA depends upon the site *CSMS* to protect against other attacks to gain user credentials, such as password guessing or social engineering.

5.2 Reconciliation of Objectives with OPC UA Security Mechanisms

5.2.1 Overview

The following subsections of Section 5.2 reconcile the objectives that were described in Section 4.2 with the OPC UA functions. Each of the following subsections relates directly to the objective described in its corresponding subsection of Section 4.2.

Compared to the reconciliation against the threats of section 5.1, this reconciliation justifies the completeness of the OPC UA security architecture.

5.2.2 Authentication

OPC UA Applications support *Authentication* of the entities with which they are communicating as well as providing the necessary *Authentication* credentials to the other entities.

Application Authentication

As specified in the GetEndpoints and OpenSecureChannel services in Part 4, OPC UA client and server applications identify and authenticate themselves with http://tools.ietf.org/html/rfc3174 *X.509 Certificate*s (see [X509]). Some choices of the Communication Stack require these certificates to represent the machine or user instead of the application.

User Authentication

As described in the OpenSecureChannel service in Part 4, the OPC UA client accepts a User Identity Token from the user and passes it to the OPC UA server. The OPC UA server authenticates the user token. *OPC UA Applications* accept tokens in any of the following three forms: username/password, an X.509v3 certificate (see [X509]) or a WS-SecurityToken

As specified in the sections of the CreateSession and ActivateSession services in Part 4, if the User Identity Token is a *Digital Certificate* then this token is validated with a challenge-response process. The server provides a *Nonce* and signing algorithm as the challenge in its CreateSession response. The client responds to the challenge by signing the server's *Nonce* and providing it as an argument in its subsequent ActivateSession call.

5.2.3 Authorization

OPC UA does not specify how user or client *Authorization* is to be provided. *OPC UA Applications* that are part of a larger industrial automation product may manage *Authorizations* consistent with the *Authorization* management of that product. Identification and *Authentication* of users is specified in OPC UA so that client and server applications can recognize the user in order to determine the *Authorizations* of the user.

OPC UA servers respond with the Bad_UserAccessDenied error code to indicate an *Authorization or Authentication* error as specified in the status codes section of Part 4.

5.2.4 Confidentiality

OPC UA uses *Symmetric* and *Asymmetric Encryption* to protect *Confidentiality as a security objective*. Thereby *Asymmetric Encryption* is used for key agreement and *Symmetric Encryption* for securing all other messages sent between OPC UA applications. Encryption mechanisms are specified in [UA Part 6].

OPC UA relies upon the site *CSMS* to protect *Confidentiality* on the network and system infrastructure. OPC UA relies upon the *PKI* to manage keys used for *Symmetric* and *Asymmetric Encryption*.

5.2.5 Integrity

OPC UA uses *Symmetric* and *Asymmetric Signatures* to address *Integrity* as a security objective. The *Asymmetric Signatures* are used in the key agreement phase during the *Secure Channel* establishment. The *Symmetric Signatures* are applied to all other messages.

OPC UA relies upon the site *CSMS* to protect *Integrity* on the network and system infrastructure. OPC UA relies upon the *PKI* to manage keys used for *Symmetric* and *Asymmetric Signatures*.

5.2.6 Auditability

As specified in the UA *Auditing* section of Part 4, OPC UA supports audit logging by providing traceability of activities through the log entries of the multiple clients and servers that initiate, forward, and handle the activity. OPC UA depends upon *OPC UA Application* products to provide an effective audit logging scheme or an efficient manner of collecting the audit events of all nodes. This scheme may be part of a larger industrial automation product of which the *OPC UA Applications* are a part.

5.2.7 Availability

OPC UA minimizes the impact of message flooding as described in Section 5.1.2.

Some attacks on *Availability* involve opening more sessions than a server can handle thereby causing the server to fail or operate poorly. Servers reject sessions that exceed their specified maximum number.

6 Implementation considerations

This section provides guidance to vendors that implement *OPC UA Application*s. Since many of the countermeasures required to address the threats described above fall outside the scope of the OPC UA specification, the advice in this section suggests how some of those countermeasures should be provided.

For each of the following areas, this section defines the problem space, identifies consequences if appropriate countermeasures are not implemented and recommends best practices.

6.1 Appropriate Timeouts:

Timeouts, the time that the implementation must wait (usually for an event such as message arrival), play a very significant role in influencing the security of an implementation. Potential consequences include

- Denial of service: Denial of service conditions may exist when a client does not reset a session, if the timeouts are very large.
- Resource consumption: When a client is idle for long periods of time, the server must keep the client information for that period, leading to resource exhaustion.

The implementer should use reasonable timeouts for each connection stage.

6.2 Strict Message Processing

The specifications often specify the format of the right messages and are silent on what the implementation should do for messages that deviate from the specification. Typically, the implementations continue to parse such packets, leading to vulnerabilities.

- The implementer should do strict checking of the message format and should either drop the packets or send an error message as described below.

- Error handling uses the error code, defined in Part 4, which most precisely fits the condition.

6.3 Random Number Generation

Random numbers that meet security needs can be generated by suitable functions that are provided by cryptography libraries. Common random function such as using rand() provided by the CRT doesn't generate enough entropy. As an alternative, implementers could use the random number generator provided by the MS Windows Crypto library (WinCrypt library) or by OpenSSL.

6.4 Special and Reserved Packets

The implementation must understand and correctly interpret any message types that are reserved as special (such as broadcast and multicast addresses in IP specification). Failing to understand and interpret those special packets may lead to vulnerabilities.

6.5 Rate Limiting and Flow Control

OPC-UA does not provide rate control mechanisms, however an implementation can incorporate rate control.