IEC TR 62541-2

Edition 2.0 2016-10

TECHNICAL REPORT

colour inside

OPC unified architecture – Part 2: Security Model

IEC TR 62541-2:2016-10(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing 20 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
65 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IEC TR 62541-2

Edition 2.0 2016-10

TECHNICAL REPORT

colour inside

OPC unified architecture – Part 2: Security Model

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.100.01

ISBN 978-2-8322-3641-3

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

OPC UNIFIED ARCHITECTURE –

Part 2: Security Model

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC TR 62541-2, which is a technical report, has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this technical report is based on the following documents:

Enquiry draft	Report on voting
65E/413/DTR	65E/464/RVC

Full information on the voting for the approval of this technical report can be found in the report on voting indicated in the above table.

This second edition cancels and replaces the first edition of IEC TR 62541-2, published in 2010.

This second edition includes no technical changes with respect to the first edition but a number of clarifications and additional text for completeness.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

Throughout this document and the referenced other parts of the series, certain document conventions are used:

– Italics are used to denote a defined term or definition that appears in the “Terms and definition” clause in one of the parts of the series.
– Italics are also used to denote the name of a service input or output parameter or the name of a structure or element of a structure that are usually defined in tables.
– The italicized terms and names are also often written in camel-case (the practice of writing compound words or phrases in which the elements are joined without spaces, with each element's initial letter capitalized within the compound). For example the defined term is AddressSpace instead of Address Space. This makes it easier to understand that there is a single definition for AddressSpace, not separate definitions for Address and Space.

A list of all parts of the IEC 62541 series, published under the general title *OPC unified architecture*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

OPC UNIFIED ARCHITECTURE –

Part 2: Security Model

1 Scope

This part of IEC 62541, which a Technical Report, 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 provides definition of common security terms that are used in this and other parts of the OPC UA specification. It gives an overview of the security features that are specified in other parts of the OPC UA specification. It references services, mappings, and *Profile*s that are specified normatively in other parts of this multi-part specification. It provides suggestions or best practice guidelines on implementing security. Any seeming ambiguity between this part of IEC 62541 and one of the normative parts of IEC 62541 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 developers look into all aspects of security and decide how they can be addressed in the application.

This part of IEC 62541 is directed to readers who will develop OPC UA *Client* or *Server* applications or implement the OPC UA services layer. It is also for end users that wish to understand the various security features and functionality provided by OPC UA. It also offers some suggestions that can be applied when deploying systems. These suggestions are generic in nature since the details would depend on the actual implementation of the *OPC UA Application*s and the choices made for the site security.

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 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62351 (all parts), *Power systems management and associated information exchange – Data and communications security*

IEC TR 62541-1, *OPC unified architecture – Part 1: Overview and concepts*

IEC 62541-4, *OPC unified architecture – Part 4: Services*

IEC 62541-5, *OPC unified architecture – Part 5: Information Model*

IEC 62541-6, *OPC unified architecture – Part 6: Mappings*

IEC 62541-7, *OPC unified architecture – Part 7: Profiles*

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

Available from Internet: http://www.w3.org/TR/soap12-part1/ (website checked 2016-04-05)

SOAP Part 2: SOAP Version 1.2 Part 2: Adjuncts

Available from Internet: http://www.w3.org/TR/soap12-part2/ (website checked 2016-04-05)

XML Encryption: XML Encryption Syntax and Processing

Available from Internet: http://www.w3.org/TR/xmlenc-core/ (website checked 2016-04-05)

XML Signature:: XML-Signature Syntax and Processing

Available from Internet: http://www.w3.org/TR/xmldsig-core/ (website checked 2016-04-05)

WS Security: SOAP Message Security 1.1

Available from Internet: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (website checked 2016-04-05)

WS Secure Conversation: Web Services Secure Conversation Language (WS-SecureConversation)

Available from Internet:http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf (website checked 2016-04-05)

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

Available from Internet: http://www.ietf.org/rfc/rfc2246.txt (website checked 2016-04-05)

:X509: X.509 Public Key Certificate Infrastructure

Available from Internet: https://www.ietf.org/rfc/rfc2459 (website checked 2016-04-05)

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

Available from Internet: http://www.ietf.org/rfc/rfc2616.txt (website checked 2016-04-05)

HTTPS: RFC 2818: HTTP Over TLS

Available from Internet: http://www.ietf.org/rfc/rfc2818.txt (website checked 2016-04-05)

IS Glossary: Internet Security Glossary

Available from Internet: http://www.ietf.org/rfc/rfc2828.txt (website checked 2016-04-05)

NIST 800-57: Part 3: Application-Specific Key Management Guidance

Available from Internet:http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_PART3_key-management_Dec2009.pdf (website checked 2016-04-05)

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

Available from Internet: http:// www.nerc.com/files/cip-002-1.pdf (website checked 2016-04-05)

SHA-1: Secure Hash Algorithm RFC

Available from Internet: http://tools.ietf.org/html/rfc3174 (website checked 2016-04-05)

PKI: Public Key Infrastructure article in Wikipedia

Available from Internet: http://en.wikipedia.org/wiki/Public_key_infrastructure (website checked 2016-04-05)

X509 PKI: Internet X.509 Public Key Infrastructure

Available from Internet: http://www.ietf.org/rfc/rfc3280.txt (website checked 2016-04-05)

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TR 62541-1 as well as the following apply.

3.1.1
Application Instance
individual installation of a program running on one computer

Note 1 to entry: There can be several Application Instances of the same application running at the same time on several computers or possibly the same computer.

3.1.2
Application Instance Certificate
Digital Certificate of an individual *Application Instance* that has been installed in an individual host

Note 1 to entry: Different installations of one software product would have different Application Instance *Certificates*.

3.1.3
Asymmetric Cryptography
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 1 to entry: Asymmetric Cryptography is 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 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 according to IS Glossary.

3.1.4
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*

3.1.5
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*

3.1.6
Auditability
security objective that assures that any actions or activities in a system can be recorded

3.1.7
Auditing
the tracking of actions and activities in the system, including security related activities where the *Audit* records can be used to review and verify system operations

3.1.8
Authentication
security objective that assures that the identity of an entity such as a *Client*, *Server*, or user can be verified

3.1.9
Authorization
the ability to grant access to a system resource

3.1.10
Availability
security objective that assures that the system is running normally; that is, no services have been compromised in such a way to become unavailable or severely degraded

3.1.11
CertificateAuthority
entity that can issue *Digital Certificate*s, also known as a CA

Note 1 to entry: The *Digital Certificate* certifies the ownership of a Public Key by the named subject of the *Certificate*. This allows others (relying parties) to rely upon signatures or assertions made by the Private Key that corresponds to the *Public Key* that is certified. In this model of trust relationships, a CA is a trusted third party that is trusted by both the subject (owner) of the *Certificate* and the party relying upon the *Certificate*. CAs are characteristic of many Public Key infrastructure (PKI) schemes.

3.1.12
CertificateStore
persistent location where *Certificate*s and *Certificate* revocation lists (CRLs) are stored

Note 1 to entry: It may be a disk resident file structure or on Windows platforms, it may be a Windows registry location.

3.1.13
Confidentiality
security objective that assures the protection of data from being read by unintended parties

3.1.14
Cryptography
transforming clear, meaningful information into an enciphered, unintelligible form using an algorithm and a key

3.1.15
Cyber Security Management System
CSMS
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

3.1.16
Digital Certificate
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*

3.1.17
Digital Signature
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*

3.1.18
Hash Function
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) , see IS Glossary

3.1.19
Hashed Message Authentication Code
HMAC
MAC that has been generated using an iterative *Hash Function*

3.1.20
Integrity
security objective that assures that information has not been modified or destroyed in an unauthorized manner, see IS Glossary

3.1.21
Key Exchange Algorithm
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 1 to entry: A typical example of a Key Exchange Algorithm is the SSL Handshake Protocol specified in SSL/TLS.

3.1.22
Message Authentication Code
MAC
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

3.1.23
Message Signature
Digital Signature used to ensure the *Integrity* of *Messages* that are sent between two entities

Note 1 to entry: There are several ways to generate and verify Message Signatures however they can be categorized as symmetric (See 3.1.34) and asymmetric (See 3.1.5) approaches.

3.1.24
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

3.1.25
Nonce
random number that is used once, typically by algorithms that generate security keys

3.1.26
OPC UA Application
OPC UA *Client*, which calls OPC UA services, or an OPC UA *Server*, which performs those services

3.1.27
Private Key
the secret component of a pair of cryptographic keys used for *Asymmetric Cryptography*

3.1.28
Public Key
the publicly-disclosed component of a pair of cryptographic keys used for *Asymmetric Cryptography*, see IS Glossary

3.1.29
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 1 to entry: 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 according to IS Glossary. See PKI and X509 PKI for more details on Public Key Infrastructures.

3.1.30
Rivest-Shamir-Adleman
RSA
algorithm for *Asymmetric Cryptography*, invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman, see IS Glossary

3.1.31
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

3.1.32
Symmetric Cryptography
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), see IS Glossary

3.1.33
Symmetric Encryption
the mechanism used by *Symmetric Cryptography* for encrypting and decrypting data with a cryptographic key shared by two entities

3.1.34
Symmetric Signature
the mechanism used by *Symmetric Cryptography* for signing data with a *cryptographic key* shared by two entities

Note 1 to entry: 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. Definition 3.1.19 defines a typical example for an algorithm that generates Symmetric Signatures.

3.1.35
TrustList
list of *Certificate*s that an application has been configured to trust

3.1.36
Transport Layer Security
TLS
standard protocol for creating *Secure Channel*s over IP based networks

3.1.37
X.509 Certificate
Digital Certificate in one of the formats defined by X.509 v1, 2, or 3

Note 1 to entry: An X.509 Certificate contains a sequence of data items and has a Digital Signature computed on that sequence.

3.2 Abbreviations

AES Advanced Encryption Standard
CA Certificate Authority
CRL Certificate Revocation List
CSMS Cyber Security Management System
DNS Domain Name System
DSA Digital Signature Algorithm
ECDH Elliptic Curve Diffie-Hellman
ECDSA Elliptic Curve Digital Signature Algorithm
HMAC Hash-based Message Authentication Code
NIST National Institute of Standard and Technology
PKI Public Key Infrastructure
RSA public key algorithm for signing or encryption, Rivest, Shamir, Adleman
SHA Secure Hash Algorithm (Multiple versions exist SHA1, SHA256,…)
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

3.3 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 *Client*s and *Servers* are on the same host and can be more easily protected from external attack. Some *Client*s 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.

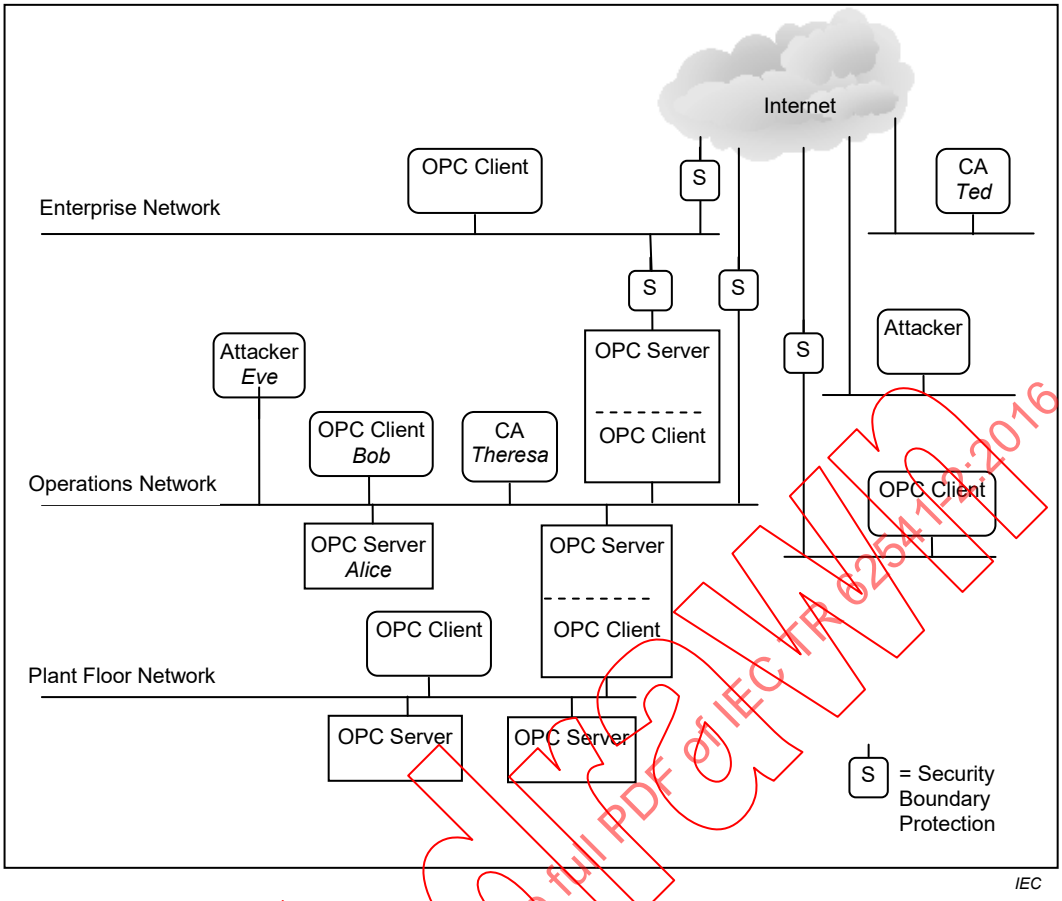

Figure 1 – OPC UA network model

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 4.2.2, 4.2.3, 4.2.4, 4.2.5, 4.2.6, and 4.2.7, and Subclause 5.2 reconciles these objectives against the OPC UA functions. Clause 6 offers additional best practice guidelines to *Client* and *Server* developers or those that deploy *OPC UA Applications*.

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 access 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 shall 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 along with *Authentication* and *Authorization* mechanisms for accessing that secret.

4.2.5 Integrity

Receivers shall receive the same information that the original 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 (successful actions are tracked),
- to identify the initiator of certain actions (user activity is tracked),
- that attempts to compromise the system were denied (unsuccessful actions are tracked).

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. Subclauses 4.3.2, 4.3.3, 4.3.4, 4.3.5, 4.3.6, 4.3.7, 4.3.8, 4.3.9, 4.3.10, and 4.3.11 list the currently known threats to environments in which OPC UA will be deployed. Following Clause 4, that describes the OPC UA security architecture and functions, Subclause 5.1 reconciles these threats against the OPC UA functions.

4.3.2 Message flooding

An attacker can send a large volume of *Message*s, 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 *Message*s. 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 *Message*s in order to exhaust the *Server*'s resources.

In general, *Message* flooding may impair the ability to communicate with an OPC UA entity and result in denial of service.

Message flooding impacts *Availability*.

See 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 *Message*s. 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 5.1.3 for the reconciliation of this threat.

4.3.4 Message spoofing

An attacker may forge *Message*s from a *Client* or a *Server*. Spoofing may occur at multiple layers in the protocol stack.

By spoofing *Message*s from a *Client* or a *Server*, attackers may perform unauthorized operations and avoid detection of their activities.

Message spoofing impacts Integrity and Authorization.

See 5.1.4 for the reconciliation of this threat.

4.3.5 Message alteration

Network traffic and application layer *Message*s may be captured, modified, and the modified *Message* forwarded to OPC UA *Client*s and *Server*s. *Message* alteration may allow illegitimate access to a system.

Message alteration impacts Integrity and Authorization.

See 5.1.5 for the reconciliation of this threat.

4.3.6 Message replay

Network traffic and valid application layer *Message*s may be captured and resent to OPC UA *Client*s and *Server*s at a later stage without modification. An attacker could misinform the user or send a valid command such as opening a valve but at an improper time, so as to cause damage or property loss.

Message replay impacts *Authorization*.

See 5.1.6 for the reconciliation of this threat.

4.3.7 Malformed Messages

An attacker can craft a variety of *Message*s with invalid *Message* structure (malformed XML, SOAP, UA Binary, etc.) or data values and send them to OPC UA *Client*s or *Server*s.

The OPC UA *Client* or *Server* may incorrectly handle certain malformed *Message*s 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 *Message*s as a pre-step for a multi-level attack to gain access to the underlying system of an *OPC UA Application*.

Malformed *Message*s impact *Integrity* and *Availability*.

See 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 *Message*s 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 security objectives indirectly.

See 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 *Message*s (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 security objectives.

See 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 5.1.10 for the reconciliation of this threat.

4.3.11 Compromising user credentials

An attacker obtains user credentials such as usernames, passwords, *Certificate*s, 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 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 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 Clause 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 *Client*s or *Server*s 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 *Client*s and *Server*s 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 *Client*s and *Server*s 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 4.3. The OPC Foundation recommends that *Client* and *Server* developers address the remaining threats, as detailed in Clause 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 IEC 62541-6, 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.

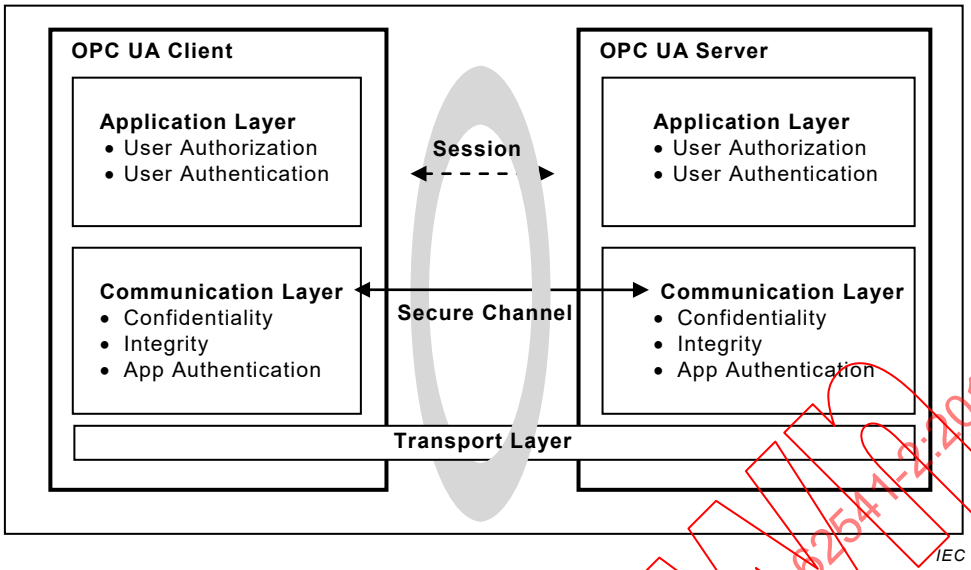

Figure 2 – OPC UA security architecture

The routine work of a *Client* application and a *Server* application to transmit 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 IEC 62541-4. 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 a user to take ownership of an existing session.

When a *Secure Channel* breaks, the *Session* will remain valid and the *Client* will 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 4.11) that is used to secure the communication between a *Client* and a *Server*. The *Secure Channel* provides encryption to maintain *Confidentiality*, *Message Signatures* 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 IEC 62541-4.

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 IEC 62541-6 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 alternative stack mappings that can be used as the Communication Layer. These mappings are described in IEC 62541-6.

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 IEC 62541-6.

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 IEC 62541-6.

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 a new connection, the Communication Layer is responsible for re-establishing the Transport Layer connection without interrupting the logical *Secure Channel*.

4.6 SecurityPolicies

A *SecurityPolicy* specifies which security mechanisms are to be used and are derived from a Security *Profile* (see 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 *SecurityPolicies* to be used for the *Secure Channel* it wishes to open. The *SecurityPolicies* specified include the following:

- algorithms for signing and encryption,
- algorithm for key derivation.

The choice of SecurityPolicy is normally made by the administrator, typically when the *Client* and *Server* products are installed. The available security policies are specified in IEC 62541-7. The Administrator can at a later date also change or modify the selection of *SecurityPolicies* as circumstances dictate.

The announcement of security policies is handled by special discovery services specified in IEC 62541-4. More details about the discovery mechanisms and policy announcement strategies can be found in IEC 62541-12.

If a *Server* serves multiple *Client*s, it maintains separate policy selections for the different *Client*s. This allows a new *Client* to select policies independent of the policy choices that other *Client*s have selected for their *Secure Channel*s.

Since computing power increases every year, 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* and to be able to migrate forward in such a case. NIST or other agencies even make predictions about the expected lifetime of algorithms (see NIST 800-57). The list of supported security policies will be updated based on recommendation such as what is published by NIST. From a deployment point of view, it is important that the periodic site review checks that the currently selected list of security profiles still fulfil the required security objectives and if they do not, then a newer selection of *Security Profiles* is selected.

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 *Client*s and *Server*s should be designed in a way that it is possible to update or add additional cryptographic algorithms to the application with little or no coding changes.

IEC 62541-7 specifies several policies which are identified by a specific unique URI. To improve interoperability among vendors' products, *Server* products shall implement these policies rather than define their own. *Client*s shall support the same policies.

4.7 Security Profiles

OPC UA *Client* and *Server* products are certified against *Profiles* that are defined in IEC 62541-7. 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. IEC 62541-7, 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, they are described in the *Profiles* that it supports.

These policies are included in Certification Testing associated with OPC UA *Client* and *Servers*. The Certification Testing ensures that the standard is followed and that the appropriate security algorithms are supported.

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. *Profile*s exist to indicate the support of user credentials to restrict or control access to data.

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 IEC 62541-4. 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 the *Certificate* and based on this check it accepts or rejects the request or response *Message* from the sender. This trust check is accomplished using the concept of *TrustLists*. *TrustLists* are a *CertificateStore* designated by an administrator. An administrator shall determine if the *Certificate* is signed, validated and trustworthy before placing it in a *TrustList*. *TrustLists* usually also include *Certificate* Revocation Lists (CRLs). OPC UA makes use of these industry standard concepts as defined by other organizations.

In OPC UA HTTPS can be used to create *Secure Channels*, however, these channels do not provide application *Authentication*. If *Authentication* is required, it shall be based on user credentials.

More details on *Application Authentication* can be found in IEC 62541-4.

4.11 OPC UA security related Services

The OPC UA Security Services are a group of abstract service definitions specified in IEC 62541-4 that are used for applying various security mechanisms to communication between OPC UA *Client*s and *Server*s.

The Discovery Service Set (specified in IEC 62541-4) defines services used by an OPC UA *Client* to inform itself about the security policies (see 4.6) and the *Digital Certificate*s of specific OPC UA *Server*s.

The services of the *Secure Channel* Service Set (specified in IEC 62541-4) 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 Certificates* 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 OPC UA *Server* is used to derive a set of cryptographic keys that are used for securing all further *Messages*. Furthermore all other service *Message*s 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 *Clients* and *Servers* 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 *Message*s to use to determine what the keys are.

4.12 Auditing

4.12.1 General

*Client*s and *Servers* generate audit records of successful and unsuccessful connection attempts, results of security option negotiations, configuration changes, system changes, user interactions 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 *Client*s capable of processing and logging them. See IEC 62541-4 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*. IEC 62541-5 defines the data types for these parameters. Other information models may extend the audit definitions. IEC 62541-7 defines *Profile*s, 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 shall 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 Technical Report.

In addition, the information in an audit record may contain sensitive or private information, thus the ability to subscribe for *Audit Events* shall be restricted to appropriate users and/or applications. As an alternative, the fields with sensitive or private information can instead contain an error code indicating access denied for users that do not have appropriate rights.

Subclauses 4.12.2, 4.12.3, 4.12.4, and 4.12.5 illustrate the behaviour of OPC UA *Servers* and *Client*s that support *Auditing*.

4.12.2 Single Client and Server

Figure 3 illustrates the simple case of a *Client* communicating with a *Server*.

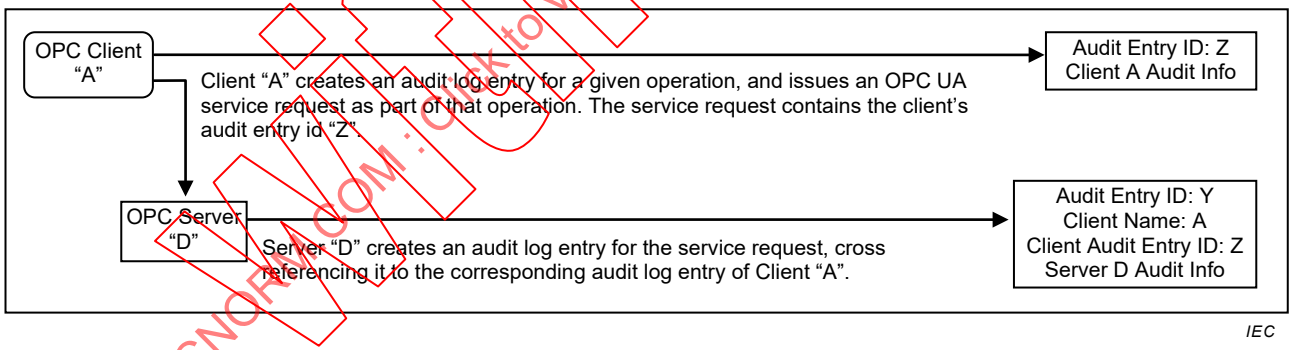

Figure 3 – Simple Servers

In this case, OPC 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*.

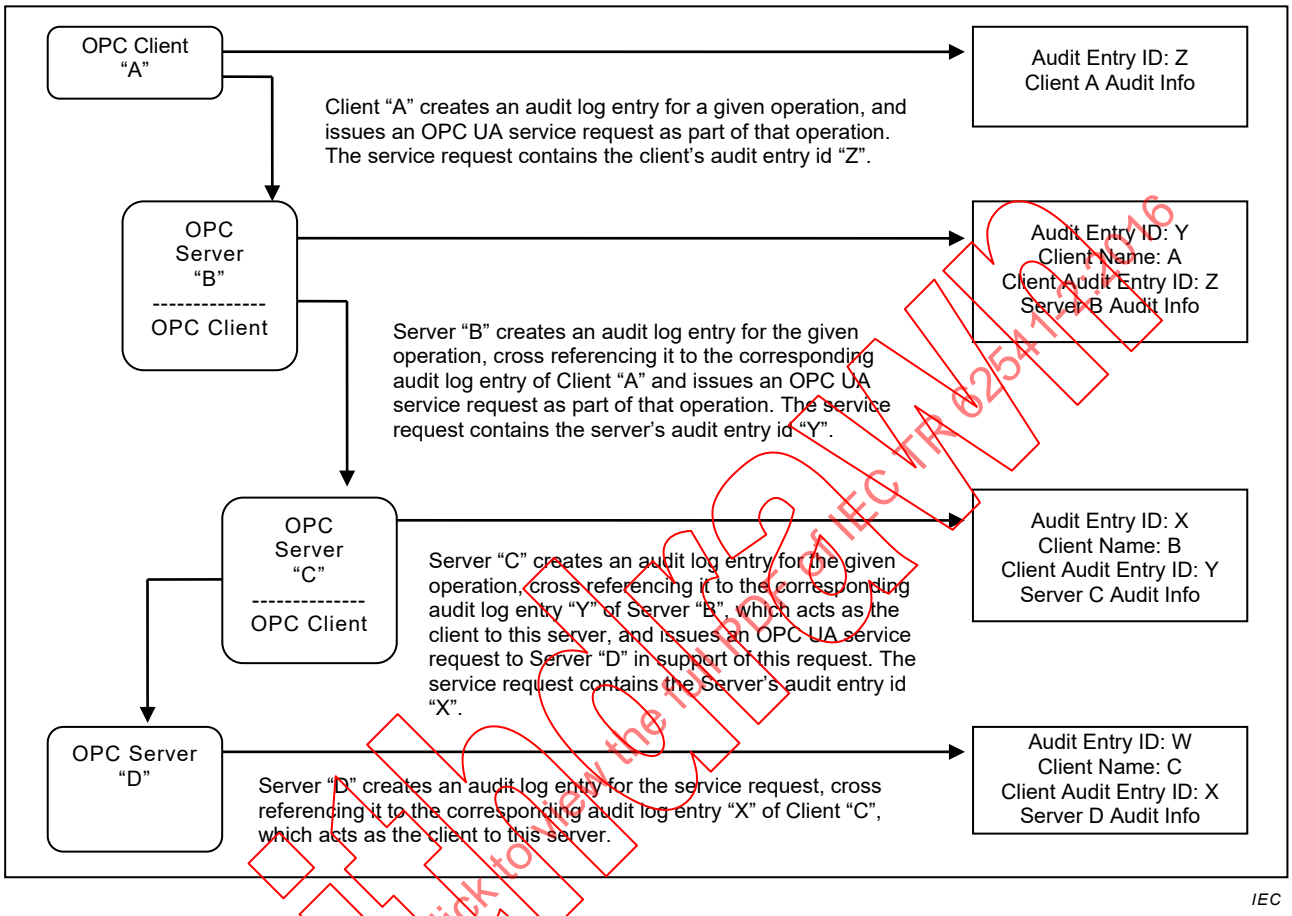

Figure 4 – Aggregating Servers

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 *Events* 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*.

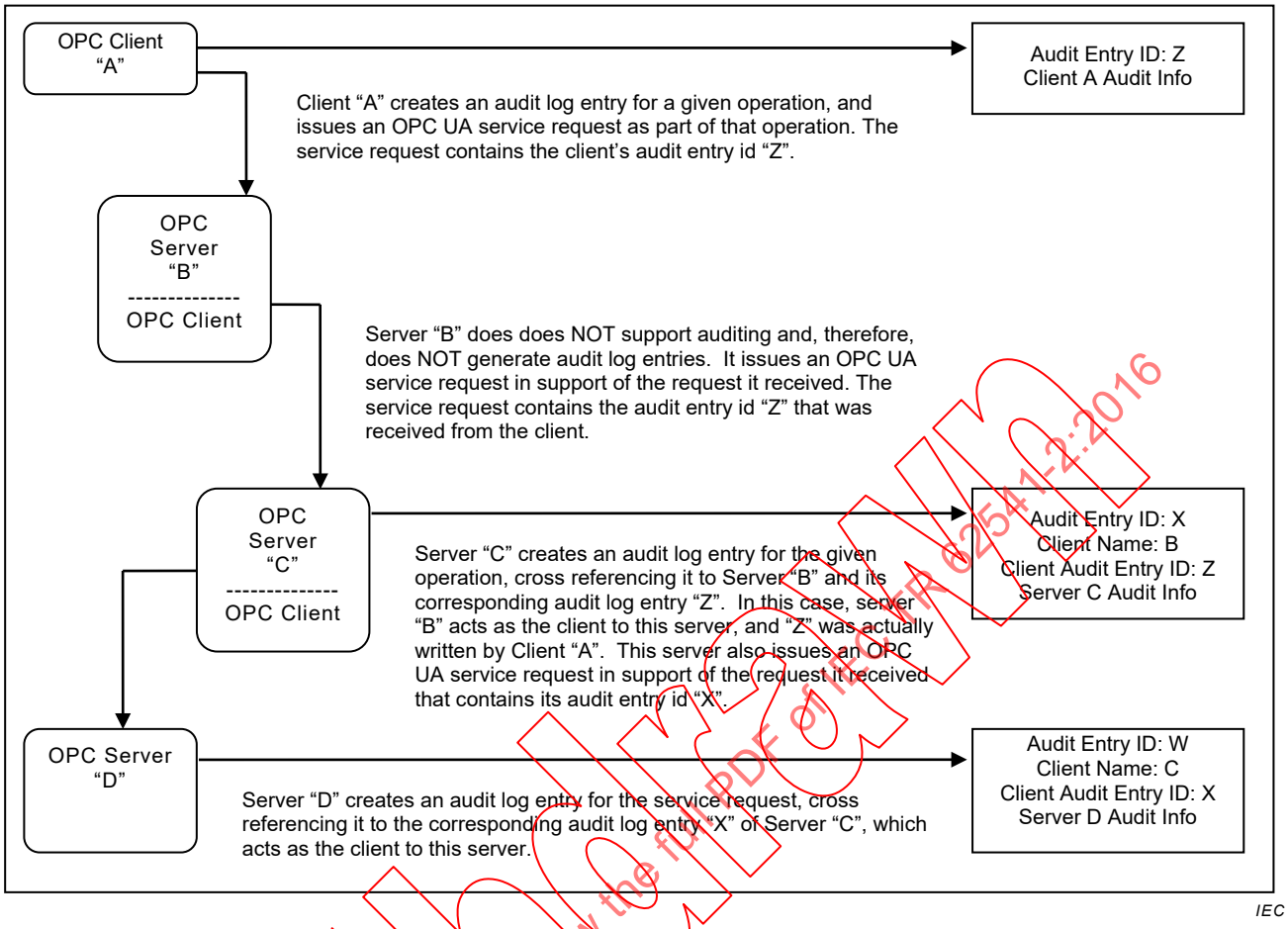

Figure 5 – Aggregation with a non-auditing Server

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*.

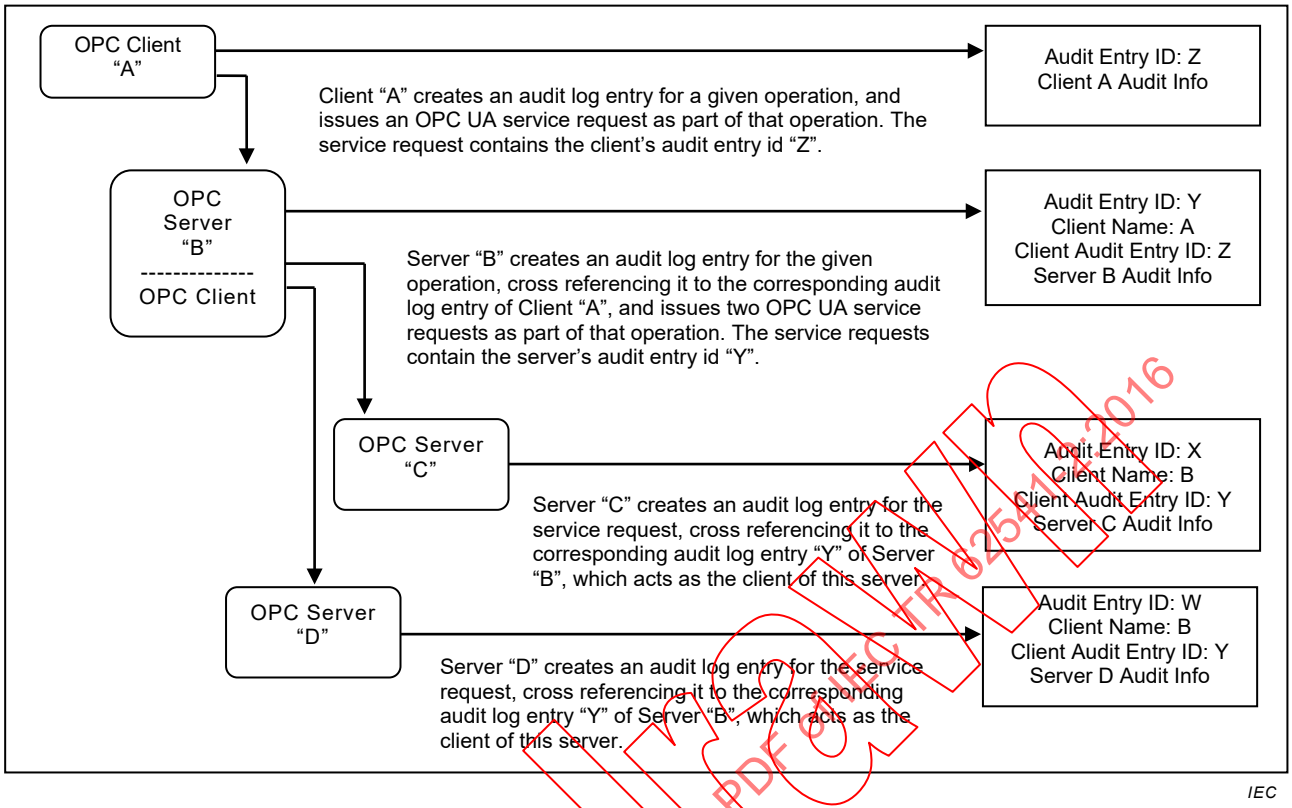

Figure 6 – Aggregate Server with service distribution

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

Subclauses 5.1.2, 5.1.3, 5.1.4, 5.1.5, 5.1.6, 5.1.7, 5.1.8, 5.1.9, 5.1.10, and 5.1.11 reconcile the threats that were described in 4.3 against the OPC UA functions. Compared to the reconciliation with the objectives that will be given in 5.2, this is a more specific reconciliation that relates OPC UA security functions to specific threats.

5.1.2 Message flooding

See 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 IEC 62541-4) and OpenSecureChannel (specified in IEC 62541-4) are the only services that the *Server* handles before the *Client* is authenticated. 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 cannot 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 more than some minimum number of bad OpenSecureChannel requests. It should also issue an alarm to alert plant personnel that an attack is underway that could 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 IEC 62541-7.

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 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 4.12). As a best practice, *Audit Events* should be monitored for excessive connection requests.

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 4.3.3 for a description of this threat.

OPC UA provides encryption to protect against eavesdropping as described in 5.2.5.

5.1.4 Message spoofing

See 4.3.4 for a description of this threat.

As specified in IEC 62541-4 and IEC 62541-6, OPC UA counters *Message* spoofing threats by providing the ability to sign *Messages*. Additionally *Messages* will always contain a valid Session ID, *Secure Channel* ID, Request ID, Timestamp as well as the correct sequence number.

5.1.5 Message alteration

See 4.3.5 for a description of this threat.

OPC UA counters *Message* alteration by the signing of *Messages* that are specified in IEC 62541-4. If *Messages* are altered, checking the signature will reveal any changes and allow the recipient to discard the *Message*. This check can also prevent unintentional *Message* alteration due to communication transport errors.

5.1.6 Message replay

See 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 IEC 62541-4 and IEC 62541-6).

5.1.7 Malformed Messages

See 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. Invalid *Messages* are discarded. This is specified in IEC 62541-4 and IEC 62541-6.

5.1.8 Server profiling

See 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 IEC 62541-4.

5.1.9 Session hijacking

See 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 IEC 62541-4. Hijacking a *Session* would thus first require compromising the security context.

5.1.10 Rogue Server

See 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 4.3.11 for a description of this threat.

OPC UA protects user credentials sent over the network by encryption as described in 5.2.5.

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

Subclauses 5.2.2, 5.2.4, 5.2.5, 5.2.6, 5.2.7, and 5.2.8 reconcile the objectives that were described in 4.2 with the OPC UA functions.

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

5.2.2 Application Authentication

OPC UA Applications support *Authentication* of the entities with which they are communicating. As specified in the GetEndpoints and OpenSecureChannel services in IEC 62541-4, OPC UA *Client* and *Server* applications identify and authenticate themselves with *X.509 Certificate*s (see [:X509]). Some choices of the Communication Stack require these *Certificate*s to represent the machine or user instead of the application.

5.2.3 User Authentication

OPC UA Applications support *Authentication* of users by providing the necessary *Authentication* credentials to the other entities. As described in the OpenSecureChannel service in IEC 62541-4, the OPC UA *Client* accepts a UserIdentityToken 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 CreateSession and ActivateSession services in IEC 62541-4, if the UserIdentityToken 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.

5.2.4 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 *Authorization* level 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 defined in IEC 62541-4.

5.2.5 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.

5.2.6 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 *Message*s.

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*.

5.2.7 Auditability

As specified in the UA *Auditing* description in IEC 62541-4, OPC UA supports *Audit* logging by providing traceability of activities through the log entries of the multiple *Client*s 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.

5.2.8 Availability

OPC UA minimizes the impact of *Message* flooding as described in 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. Other aspects of OPC UA such as OPC UA Secure Conversation or WS Secure Conversation can also affect availability and are discussed in IEC 62541-6.

6 Implementation and deployment considerations

6.1 Overview

Clause 6 provides guidance to vendors that implement *OPC UA Applications*. Since many of the countermeasures required to address the threats described above fall outside the scope of the OPC UA specification, the advice in Clause 6 suggests how some of those countermeasures should be provided.

For each of the following areas, Clause 6 defines the problem space, identifies consequences if appropriate countermeasures are not implemented and recommends best practices.

6.2 Appropriate timeouts

Timeouts, the time that the implementation shall 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* shall keep the *Client*'s buffered *Message* or information for that period, leading to resource exhaustion.

The implementer should use reasonable timeouts for each connection stage.

6.3 Strict Message processing

The specifications often specify the format of the right *Message*s and are silent on what the implementation should do for *Message*s 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 IEC 62541-4, which most precisely fits the condition.
- All arrays lengths and string lengths should be strictly enforced and processed.

6.4 Random number generation

Random numbers that meet security needs can be generated by suitable functions that are provided by cryptography libraries. Common random functions such as using rand() provided by the "C" standard library do not generate enough entropy. As an alternative, implementers could use the random number generator provided by the Microsoft Windows Crypto library (WinCrypt library) or by OpenSSL.

6.5 Special and reserved packets

The implementation shall 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.

6.6 Rate limiting and flow control

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

6.7 Administrative access

OPC UA describes that certain functionality, such as the management of *CertificateStores*, should be restricted to administrators. This Multi-part standard does not describe the details associated with administrative access. The nature of administrative access varies from platform to platform. Some platforms only have a single administrator. Other platforms provide multiple levels of administrative access such as backup administrator, network administrator, configuration administrator etc. The deployment site should make appropriate selections for administrator access and the implementer should allow for the configuration of appropriate administrator account access.

6.8 Alarm related guidance

OPC UA supports a robust *Alarm* and *Condition* information model, which includes the ability to disable alarms, shelve alarms and to generally manage alarms. Alarm processing and management is an important part of maintaining efficient control of a plant. From a security point of view it is important that this avenue be adequately protected, to ensure that a rogue agent does not create a dangerous or financial situation. OPC UA provides the tools required for this protection, but the implementer needs to ensure that they are exercised correctly. All functions that allow changes to the running environment are able to generate *Audit Events* and are to be restricted to appropriate users.

The disabling of Alarms is one such function that should be restricted to personnel with appropriate access rights. Furthermore, any action that disables an alarm, whether it be initiated by personnel or some automated system, should generate an *Audit* Event indicating the action.

The shelving of alarms should follow similar guideline as the disabling of alarms with regard to access and *Auditing*, although it may be available to a wider range of users (operators, engineers). Also the implementer should ensure that appropriate timeouts are configured for Alarm Shelving. These timeouts should ensure that an Alarm cannot be shelved for a period of time that could cause safety concerns.

Dialog *Events* could also be used to overload a *Client*. It would be a best practice for *Servers* that support dialogs to restrict the number of concurrent dialogs that could be active. Also Dialogs should include some timeout period to ensure that they are not used to create a DOS. *Client* implementers should also ensure that any dialog processing cannot be used to overwhelm an operator. The maximum number of open dialogs should be restricted and dialogs should be able to be ignored (i.e. other processing should still be available).

6.9 Program access

OPC UA describes functionality that allows for programs to be executed as part of the OPC UA *Server*. These programs can be used to perform advanced control algorithms or other actions. The use of these actions should be restricted to personnel with appropriate access rights. Furthermore, the definition of Programs should be carefully monitored. It is recommended that statistics be maintained regarding the number of defined programs in addition to their execution frequency. This information shall be available to administrative personnel. In no case should an unlimited number of program executions be allowed.