Systerel
S2OPC
Safe & Secure OPC
Security Target CSPN

Confidentiality	Reference	Version	Number of pages	Last revision date
Non Confidential	**C838_ST_CSPN**	**F**	**26**	**20/03/2023**

	Name	Visa	Date
Written by	Vincent Monfort	[illegible]	20/03/2023
Checked by	Vincent Lacroix	[illegible]	20/03/2023
Approved by	Vincent Lacroix	[illegible]	20/03/2023

Summary

1. INTRODUCTION

1.1. Document context

This document is intended to define the target of evaluation used for the ANSSI defined CSPN security certification framework (cf [R16]) for demo applications based on S2OPC product. In the whole document, the acronym TOE (Target of Evaluation) designates the component being evaluated.

1.2. History

Version Date	Author	Chapter	Modifications
A 11/02/2020	V.Monfort	All	Creation of document. Take peer and ANSSI review into account. Major revision created from minor revision 13A.
B 07/04/20	V.Monfort	All	Take CESTI review into account. Take peer review into account. Update the demo binaries description. Major revision created from minor revision 3B.
C 08/04/21	V.Monfort V.Lacroix	All	Take pre-CSPN evaluation remarks into account Reference NIST guidelines for keys management Modification of client / server applications API Update applicable documents version Update links to API Remove “Annex D S2OPC API” since API is described in S2OPC gitlab documentation page referenced in §2.2.4. Update external library version. Update parameters needed for S2OPC demo server. Major revision created from minor revision 3C.

D 23/01/23	V.Monfort V.Lacroix	All	User applications are now included in the TOE. Update functional scope figure. Update demo server command line. Update demo clients command documentation. Add information on session user configuration and use of user in client command lines. Update reference to new version of R15. Take peer review into account. Take ANSSI review into account. Take quality review into account. Create major version from minor version 18D.
E 21/02/23	V.Lacroix	§4	Modify hypotheses to take ANSSI review into account.
F 20/03/23	V.Monfort	§1.6	Update S2OPC version to 1.3.1

1.3. Applicable documents

Doc	Title	Reference
[A1]	S2OPC Wiki	Tag S2OPC_Toolkit_1.3.0 in https://gitlab.com/systerel/S2OPC.wiki.git

1.4. Reference documents

Doc	Title	Reference
[R1]	OPC UA Overview and Concepts	OPC UA Part 1: Overview and Concepts 1.03 Specification.pdf
[R2]	OPC UA Security Model	OPC UA Part 2: Security Model 1.03 Specification.pdf
[R3]	OPC UA Address Space Model	OPC UA Part 3: Address Space Model 1.03 Specification.pdf
[R4]	OPC UA Services	OPC UA Part 4: Services 1.03 Specification.pdf
[R5]	OPC UA Information Model	OPC UA Part 5: Information Model 1.03 Specification.pdf
[R6]	OPC UA Mappings	OPC UA Part 6: Mappings 1.03 Specification.pdf
[R7]	OPC UA Profiles	OPC UA Part 7: Profiles 1.03 Specification.pdf
[R8]	OPC UA Data Access	OPC UA Part 8: DataAccess 1.03 Specification.pdf
[R9]	OPC UA Alarms and Conditions	OPC UA Part 9: Alarms and Conditions 1.03 Specification.pdf
[R10]	OPC UA Programs	OPC UA Part 10: Programs 1.03 Specification.pdf
[R11]	OPC UA Historical Access	OPC UA Part 11: Historical Access 1.03 Specification.pdf
[R12]	OPC UA Discovery	OPC UA Part 12: Discovery 1.03 Specification.pdf
[R13]	OPC UA Aggregates	OPC UA Part 13: Aggregates 1.03 Specification.pdf
[R14]	OPC UA Errata	OPC UA 1.03.1 OPC UA 1.03.2 OPC UA 1.03.3 OPC UA 1.03.7 Specification Errata.pdf
[R15]	Recommandations relatives à l'authentification multifacteur et aux mots de passe	ANSSI-PG-078 08/10/2021 (anssi-guide-authentification_multifacteur_et_mots_de_passe.pdf)
[R16]	Certification de Sécurité de Premier Niveau des produits des technologies de l'information	ANSSI-CSPN-CER-P-01/2.0
[R17]	NIST SP 800-57 Part 1 Rev.5 – Recommendation for Key Management	https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

1.5. Terms and abbreviations

ANSSI	Agence Nationale de la Sécurité des Systèmes d'Information
API	Application Programming Interface
CA	Certificate Authority
COTS	Commercial Off-The-Shelf software
CRL	Certificate Revocation List
CSPN	Certification de Sécurité de Premier Niveau
NIST	National Institute of Standards and Technology
OPC UA	Open Platform Communications Unified Architecture
ST	Security Target
TOE	Target of Evaluation

1.6. Product identification

Editor	Systerel Les Portes de l'Arbois, Bâtiment A 1090 rue René Descartes 13100 Aix-en-Provence Tél : +33 (0)4 42 90 41 20
Mail	s2opc@systerel.fr
Link	http://www.systerel.fr/
Product link	https://gitlab.com/systerel/S2OPC/
Version or tag	S2OPC_Toolkit_1.3.1

2. REFERENCE PRODUCT DESCRIPTION

2.1. Product general description

OPC UA is a platform-independent standard through which various kinds of systems and devices can communicate by sending request and response messages between clients and servers. It supports robust secure communication that assures the identity of OPC UA applications and resists attacks (cf. 5.2 General OPC Unified Architecture, [R1]).

This standard defines three data encodings for communication:

- OPC UA Binary,
- OPC UA XML,
- OPC UA JSON.

The reference product, S2OPC, is an implementation of OPC UA communication stack and services for OPC UA Binary data encoding. It can be used as a client, as a server or as a client-server. It is developed according to the specification published by the OPC Foundation (cf. [R1] to [R14]) (https://opcfoundation.org).

Note: Starting from revision 1.04 of the specification, using PubSub communication pattern instead of client/server is possible. S2OPC product provides an OPC UA PubSub implementation but it is not included in the TOE.

2.2. Product functions description

2.2.1. OPC UA secure communication

The OPC UA secure communication, called “Secure Channel” or “Secure Conversation”, is parameterized through a security mode and policy.

The standard defines three security modes (cf. [R4] and [R6]), all implemented by S2OPC product:

- None: unsecure communication,
- Sign: communication with integrity and authenticity properties,
- SignAndEncrypt: communication with integrity, authenticity and confidentiality properties.

The standard defines security policies (cf. [R7]), each one defines standard algorithms for the cryptographic operations. The S2OPC product implements five security policies:

- None: no cryptographic operations,
- Basic256,
- Basic256Sha256,
- Aes128_Sha256_RsaOaep,
- Aes256_Sha256_RsaPss.

Note: the None security policy is only available in combination with None security mode.

2.2.2. OPC UA session

In addition to the OPC UA secure communication stack, S2OPC product is also an implementation of OPC UA session service set (cf. [R4]). An activated session identifies a user. It is necessary to activate a session for all OPC UA high level services, except for OPC UA discovery services.

The OPC UA session is parameterized through a user token policy (cf. [R4]), the S2OPC product implements two user policies:

- Anonymous: user not identified,
- UserName: user identified with username/password token.

On server side, S2OPC product provides user authentication mechanism but the actual user authentication function implementation is exported to the application implementing the OPC UA server.

2.2.3. OPC UA services

The OPC UA standard defines several services sets (cf. [R4]). This section presents the services implemented by the S2OPC product. The SecureChannel and Session service sets are excluded from this section since they encapsulate the use of the following services and are defined in the previous sections 2.2.1 and 2.2.2.

2.2.3.1. Client services

On client side, S2OPC product provides API to send service requests and receive service responses directly. The application is in charge of the service treatment algorithm. Some S2OPC product additional client wrappers or client demo code are proposed to simplify call to some of the OPC UA services (cf. Annex E).

2.2.3.2. Server services

S2OPC server is compliant with Nano Embedded Device Server Profile (cf. [R7] or http://opcfoundation.org/UA-Profile/Server/NanoEmbeddedDevice) with additional services, such as subscriptions, and facets, such as security facets Basic256, Basic256Sha256, Aes128_Sha256_RsaOaep and Aes256_Sha256_RsaPss.

The available functionalities are listed in annex **[S2OPC FUNCTIONAL SCOPE]**.

In addition the services listed, S2OPC product implements an internal data structure which contains the OPC UA address space and an API is provided to the application to call locally read, write and browse services (cf. Annex A).

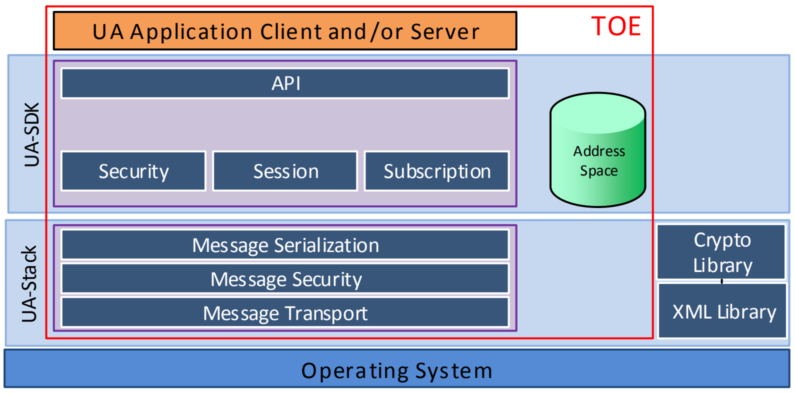
This OPC UA address space is made of nodes which are connected to each other's using references. These nodes contain attributes according to their node type (up to 22 possible attributes, see Table 12 of [R3]). Rules concerning Read/Write permission to these attributes are as follows:

- For the 13th attribute (the value), it is managed:
  - Firstly by the AccesLevel attribute,
  - Then by the UserAccesLevel (dynamically evaluated according to the user),
- For all other attributes:
  - There is no read access control policy,

- Write access control policy is associated to WriteMask and WriteMask attributes which are not implemented in S2OPC,
- As a consequence, there is a read only access on all these attributes.

Regarding the RolePermissions and UserRolePermissions attributes, they are referenced since revision 1.04 of the OPC UA specification, so not implemented in S2OPC and not part of the evaluation.

2.2.4. Product API



The S2OPC product is provided as a library (.a, .lib, .dll) that is available on the following platforms:

- Linux 32 bits and 64 bits,
- Windows 32 bits and 64 bits,
- FreeRTOS,
- Zephyr.

Note: For FreeRTOS and Zephyr, S2OPC shall be included in an image with the application and the operating system.

The S2OPC product API is defined in repository documentation page (https://systerel.gitlab.io/S2OPC/), it includes the API used by S2OPC demo Server demo and the API used by S2OPC Client demo.

The compilation and integration in an application are defined in the product Wiki (cf [A1]).

3. EVALUATION SCOPE

The TOE uses the S2OPC product described in section 2 but the evaluation is limited to some configurations.

3.1. Configuration

The secure communication configuration shall only use the security policy Basic256Sha256 (cf. C838_ST_CSPN_Annex for detailed cryptographic algorithms).

The secure communication configuration shall only use the following security modes:

- Sign: in this case no confidentiality property is provided,
- SignAndEncrypt.

Notes:

- It is necessary to only use the SignAndEncrypt security mode not to expose confidential data on the network,
- the OPC UA discovery endpoint functionality might also be deactivated to avoid exposure to unsecure secure channel connection (cf §5.4.1 of [R4]). But it is also the way for the client to get the server certificate for the secure connections and discovery endpoint functionality does not give access to the address space. For this reason the discovery endpoints are maintained in the default configuration of the TOE.

The user configuration shall only use the UserName user policy.

Note: in others words, the Anonymous user policy is still defined in the product as required by OPC UA standard but it is not activated by user configuration.

Some functionalities are not enabled by default and then outside the scope of the evaluation:

- Reverse connection mechanism,
- The services to add new nodes in server address space at runtime.

3.2. Software Dependencies

The TOE is dependent on the following libraries for binaries build and runtime (except if linked statically):

COTS Library	Revision	Patch or modification applied ?	Up to date ?	Maintained component ?
MbedTLS	**2.28.2** 14/12/2022	No	Yes	Yes
Expat (built using CMake)	**2.5.0** 25/10/2022	No	Yes	Yes

Note: These COTS libraries are external components of the TOE (see §3.4 for the scope definition).

The TOE is dependent on the following tools for binaries build only:

- Make (version 4.3),
- CMake (version 3.9.4),
- GCC (version 11.2.0),
- Python3 (version 3.6.3),
- (optional) Check (tested with libcheck version 0.14 compiled with CMake),

Note: the Check library is optional if project is built with ENABLE_TESTING variable set to 0.

3.3. Users

The users that may interact with the TOE are the following:

U1. OPC UA client: It could be a device or third-party software that initiates the communication with the TOE.

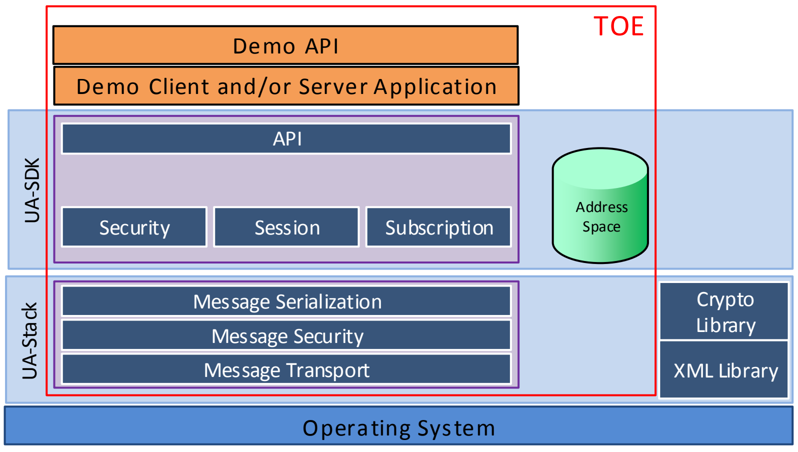
U2. OPC UA server: It could be a device or third-party software that communicates with the TOE.

3.4. Target of Evaluation

The TOE consists of demo applications based on S2OPC product. These applications are detailed in annexes [Annex D] and [Annex E].

These applications are in charge of configuring the TOE and providing user secrets in client mode or validating user secrets in server mode.

They will be evaluated on the following platform: Linux 64 bits.



The demo server API allows to:

- configure the server:
  - endpoints (URL, security),

        - address space structure, content and access control policy,
        - user authentication mechanism,
        - user authorization mechanism.
    - start the server and,
    - local OPC UA services,
    - stop the server.

The demo server waits for requests from an OPC UA client **U1** until it is stopped.

The demo client API allows to:

- establish a session through a secure channel to a server,
- send requests to an OPC UA server **U2** (one request for each type of service available on the server),
- wait for the response from the OPC UA server **U2** (the application checks the content of the response),
- close a secure channel and a session.

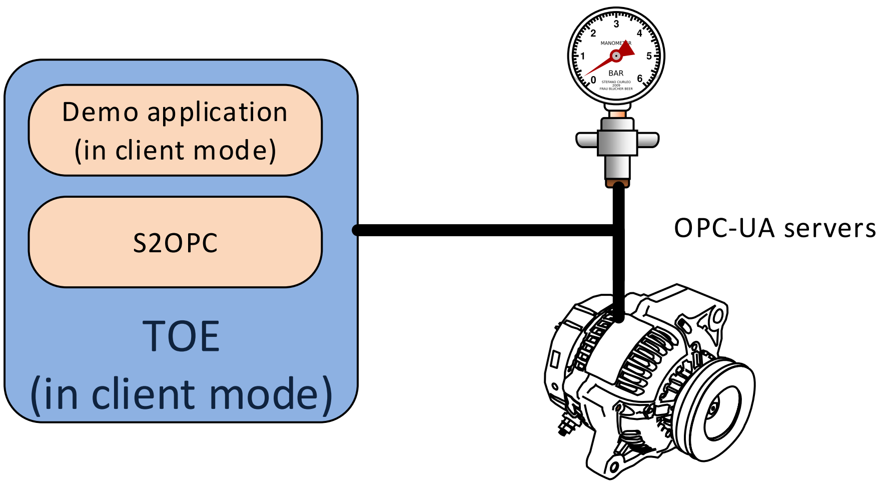
API with demo applications are defined in annexes [S2OPC Server demo API] and [S2OPC Client demo API].

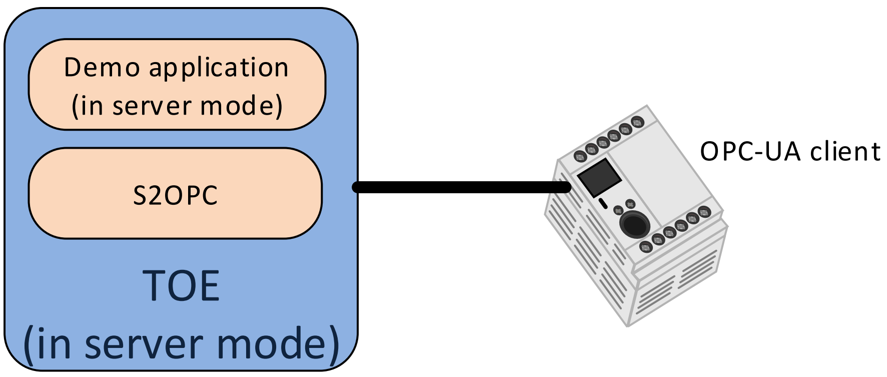
According the configuration:

- the TOE exposes one or more interfaces to OPC UA client **U1**. These interfaces correspond to endpoints according to the OPC UA standard,
- the TOE opens and closes OPC UA connections to OPC UA server **U2** according to user application requests,

These communications are built upon the TCP/IP protocol.

In order to evaluate the TOE, the following platform shall be set up by the evaluator:





Note : the client demo application can also directly communicate with the server demo application.

4. ASSUMPTIONS

Assumptions on the environment and the use case of the TOE are the following:

H1. Administrators: The administrators of the TOE are competent, trained and trustworthy.

H2. Privileges: The TOE usage does not require specific privileges in the operating system. The TOE is loaded with unprivileged user without specific capability.

H3. Operating system: The latest security updates are installed on the operating system on which the TOE is running. The TOE is loaded with a restricted access limited to trustworthy people and all applications that are loaded in the operating system are trustworthy. In particular, the attacker does not have access to this operating system.

H4. Active logging: The logging of application is operational and these logs are not corrupted.

H5. Dimensioning: The TOE and the operating system where it is loaded are properly dimensioned for theirs tasks.

H6. OPC UA user passwords: All OPC UA user passwords are compliant with ANSSI guidelines (see [R15]). This is not applicable when the anonymous OPC UA user mode is used.

H7. OPC UA certificates: All certificates are regularly updated by applying NIST guidelines (see [R17]).

H8. File system: The file system is configured to restrict access to the assets it contains. The file system users shall be configured to have the right to read, write or execute files only when it is necessary.

In particular, the rights of the TOE in the file system shall be restricted to read only when it concerns the TOE itself, the private keys and the certificates. And the rights of the TOE shall be restricted to read and write for the log files it generates whereas it shall be restricted to, at least, read only for other users.

H9. Private keys passwords: All passwords used to encrypt private keys are compliant with ANSSI guidelines (see [R15]).

5. ASSETS

The critical assets of the TOE are the following:

A1. Software: In order to work properly, the TOE software shall not be corrupted using TOE services.

A2. Private keys: the OPC UA private keys loaded in memory of the TOE and used to establish the OPC UA secure channel with the OPC UA clients **U1** or OPC UA servers **U2**.
Private keys are loaded in memory from the file system by the TOE according to the paths in its configuration. Private keys files are encrypted by a password.
Integrity, confidentiality and authenticity must be preserved.

A3. Certificates: All certificates loaded in memory by the TOE:

- The certificates of the pairs of key public/private (asset **A2**) key,
- The certificates of OPC UA servers **U2** the TOE is communicating with,
- The certificates of the CA used to authenticate the OPC UA clients **U1** and to sign the certificates of pairs of key, OPC UA public/private (asset **A2**) key, when the TOE is communicating with an OPC UA server **U2**,
- The certificates of the CA used to authenticate the OPC UA servers **U2** and to sign to the certificates of pairs of key, OPC UA public/private (asset **A2**) key, when TOE is communicating with an OPC UA client **U1**,
- The Certificate Revocation List (CRL) associated to each CA.

Certificates are loaded in memory from the file system by the TOE according to the paths in its configuration. Integrity and authenticity of these certificates must be preserved.

Note: The TOE doesn’t need OPC UA clients **U1** certificates because they are transmitted during secure channel establishment communications.

A4. Address space: the address space (cf. [R3]) is a data structure. OPC UA clients **U1** can access to address space data through an OPC UA secure channel. The application can access to the address space through API. The TOE must preserve the integrity, authenticity and confidentiality of the address space in memory.

Notes:

- Following OPCUA standard, there is no read access control policy on attributes others than values,
- The TOE does not allow writing attributes others than value attribute,
- RolePermissions and UserRolePermissions are not available in the 1.03 revision of the OPC UA specification, and therefore are not implemented in S2OPC.

A5. Encapsulated data flows: The integrity, authenticity and confidentiality of the data flows in OPC UA secure channel and sessions must be ensured.

A6. User authentication mechanism: Before accessing to the address space (asset **A4**), session users from OPC UA clients **U1** have to be authenticated. The user authentication function is provided by the application included in the TOE. The TOE must ensure the integrity and authenticity of the authentication mechanism.

A7. User credentials: The user credentials are name and passwords.

When an OPC UA client **U1** requests an access to the address space (asset **A4**), the TOE checks the user credentials to grant or refuse the authentication.

When the TOE requests an access to OPC UA server **U2**, it transmits the user credentials in order to authenticate to the server. The TOE must ensure the integrity and confidentiality of these credentials.

A8. Access control policy configuration: the TOE must ensure the integrity of the address space's access control policy configuration.

A9. User access control mechanism: Before accessing to the address space (asset **A4**), OPC UA clients **U1** have to be granted access to read and/or write address space Value attributes. The user authorization function is provided by the application included in the TOE. This function is called by S2OPC authorization mechanism. The TOE must ensure the integrity and authenticity of the authorization mechanism.

A10. Active logging: The TOE logging mechanism is operational and logs are not corrupted by TOE services.

The security requirements for the critical assets are the following:

	Asset	Confidentiality	Integrity	Availability	Authenticity
A1	Software		X		
A2	Private keys	X	X		X
A3	Certificates		X		X
A4	Address space	X	X		X
A5	Encapsulated data flows	X	X		X
A6	User authentication mechanism		X		X
A7	User credentials	X	X		
A8	Access control policy configuration		X		
A9	User access control mechanism		X		X
A10	Active logging		X	X	

6. THREATS MODEL

6.1. Attackers

The following attackers are considered:

At1. Evil device: A device plugged controlled by the attacker on the same network of the TOE.

At2. Evil OPC UA users: The attacker has compromised an OPC UA client U1 or an OPC UA server U2 with valid certificates and user credentials and tries to bypass the access control policy of the TOE.

6.2. Threats

The following threats are considered:

T1. Software alteration: The attacker manages to inject and run a corrupted software on the TOE. The code injection may be temporary or permanent and this does include any unexpected or unauthorized code execution.

T2. Private keys compromise: The attacker manages to illegally obtain a private key.

T3. Private keys alteration: The attacker manages to modify, temporary or permanently, a private key.

T4. Certificates alteration: The attacker manages to modify, temporary or permanently, a certificate.

T5. Denial of service: The attacker manages to generate a denial of service on the TOE by performing an unexpected action or by exploiting a vulnerability (sending a malformed request, using a corrupted configuration file, etc.). This denial of service can affect the whole TOE or only some of its functions.

T6. Address space compromise: The attacker manages to illegally access some parts of the address space of the TOE.

T7. Address space alteration: The attacker manages to modify, temporary or permanently, some parts of the address space of the TOE.

T8. Flows compromise: The attacker manages to fetch data by intercepting exchanges between the TOE and an external component.

T9. Flows alteration: The attacker manages to corrupt exchanges between the TOE and an external component without being detected.

T10. Authentication violation: The attacker succeeds in authenticating himself without credentials.

T11. Credentials theft: The attacker manages to steal user credentials.

T12. Credentials alteration: The attacker manages to modify user credentials.

T13. Access control violation: The attacker manages to obtain permissions that he does not normally have.

T14. Logs alteration: The attacker manages to delete or modify a local log entry without being authorized by the access control policy of the TOE.

7. SECURITY FUNCTIONS

The following security functions are considered:

SF1. Malformed input management: The TOE has been developed in order to handle correctly malformed input, in particular malformed network traffic.

SF2. Address space protection: The address space is a data structure contained in OPC UA servers. The access to address space data shall be performed with signed and encrypted OPC UA communications.

SF3. OPC UA Secure channel: The TOE supports secure connection. The secure connection allows authenticating both peers and protecting the integrity and the authenticity of exchanges. The confidentiality is also protected when SignAndEncrypt security mode is active in the secure connection.
It also guarantees non-replay of exchanges.

SF4. OPC UA Session: The TOE supports OPC UA session. The identity and the permissions of the OPC UA user are systematically checked before any action.

SF5. Secure storage of user passwords: User passwords are securely stored in the TOE in server mode. User passwords are stored as PBKDF2 hashes.

SF6. Access control policy: The access control policy of address space value attributes is strictly applied. In particular, the implementation guarantees the authenticity of privileged operations, i.e. operations that can alter identified critical assets.

Notes:

- Following OPCUA standard, there is no read access control policy on attributes others than values,
- The TOE does not allow writing attributes others than value attribute,
- RolePermissions and UserRolePermissions are not available in the 1.03 revision of the OPC UA specification, and therefore are not implemented in S2OPC.

SF7. Secure storage of private keys: Private keys are securely stored in the TOE. Private key files are encrypted using AES-256 CBC algorithm and a password which shall be given to the TOE in an interactive way at startup.

ANNEX A ASSETS VS THREATS

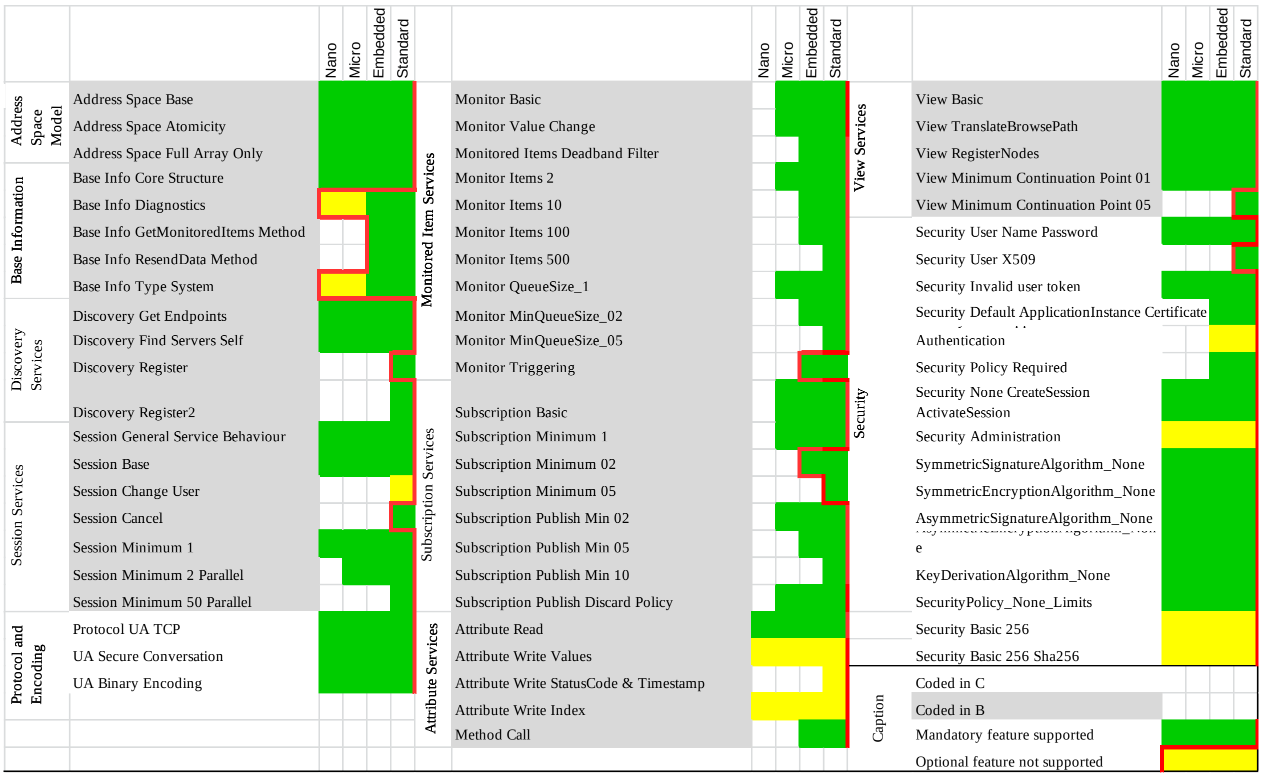
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
		Software	Private keys	Certificates	Address space	Encapsulated data flows	User authentication mechanism	User credentials	Access control policy configuration	User access control mechanism	Active logging
T1	**Software alteration**	I									
T2	**Private keys compromise**		C								
T3	**Private keys alteration**		Au I								
T4	**Certificates alteration**			Au I							
T5	**Denial of service**										Av
T6	**Address space compromise**				C						
T7	**Address space alteration**				Au I						
T8	**Flows compromise**					C					
T9	**Flows alteration**					Au I					
T10	**Authentication violation**						Au I				
T11	**Credentials theft**							C			
T12	**Credentials alteration**							I			
T13	**Access control violation**								I	Au I	
T14	**Logs alteration**										I

Au: Authenticity
Av: Availability
C: Confidentiality
I: Integrity

ANNEX B THREATS VS SECURITY FUNCTIONS

		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
		Software alteration	Private keys compromise	Privates keys alteration	Certificates alteration	Denial of service	Address space compromise	Address space alteration	Flows compromise	Flows alteration	Authentication violation	Credentials theft	Credentials alteration	Access control violation	Logs alteration
SF1	**Malformed input management**	X	X	X	X	X									X
SF2	**Address space Protection**						X	X							
SF3	**OPC UA Secure channel**								X	X					
SF4	**OPC UA Session**								X	X	X				
SF5	**Secure storage of user passwords**											X	X		
SF6	**Access control policy**													X	
SF7	**Secure storage of private keys**		X	X											

ANNEX C S2OPC FUNCTIONAL SCOPE



		Nano	Micro	Embedded	Standard
Address Space Model	Address Space Base	🟩	🟩	🟩	🟩
	Address Space Atomicity	🟩	🟩	🟩	🟩
	Address Space Full Array Only	🟩	🟩	🟩	🟩
Base Information	Base Info Core Structure	🟩	🟩	🟩	🟩
	Base Info Diagnostics	🟨	🟨	🟩	🟩
	Base Info GetMonitoredItems Method			🟩	🟩
	Base Info ResendData Method			🟩	🟩
	Base Info Type System	🟨	🟨	🟩	🟩
Discovery Services	Discovery Get Endpoints	🟩	🟩	🟩	🟩
	Discovery Find Servers Self	🟩	🟩	🟩	🟩
	Discovery Register				🟩
	Discovery Register2				🟩
Session Services	Session General Service Behaviour	🟩	🟩	🟩	🟩
	Session Base	🟩	🟩	🟩	🟩
	Session Change User				🟨
	Session Cancel				🟩
	Session Minimum 1	🟩	🟩	🟩	🟩
	Session Minimum 2 Parallel		🟩	🟩	🟩
	Session Minimum 50 Parallel				🟩
Protocol and Encoding	Protocol UA TCP	🟩	🟩	🟩	🟩
	UA Secure Conversation	🟩	🟩	🟩	🟩
	UA Binary Encoding	🟩	🟩	🟩	🟩

		Nano	Micro	Embedded	Standard
Monitored Item Services	Monitor Basic		🟩	🟩	🟩
	Monitor Value Change		🟩	🟩	🟩
	Monitored Items Deadband Filter			🟩	🟩
	Monitor Items 2		🟩	🟩	🟩
	Monitor Items 10			🟩	🟩
	Monitor Items 100			🟩	🟩
	Monitor Items 500				🟩
	Monitor QueueSize_1		🟩	🟩	🟩
	Monitor MinQueueSize_02			🟩	🟩
	Monitor MinQueueSize_05				🟩
	Monitor Triggering			🟩	🟩
Subscription Services	Subscription Basic		🟩	🟩	🟩
	Subscription Minimum 1		🟩	🟩	🟩
	Subscription Minimum 02			🟩	🟩
	Subscription Minimum 05				🟩
	Subscription Publish Min 02		🟩	🟩	🟩
	Subscription Publish Min 05			🟩	🟩
	Subscription Publish Min 10				🟩
	Subscription Publish Discard Policy		🟩	🟩	🟩
Attribute Services	Attribute Read	🟩	🟩	🟩	🟩
	Attribute Write Values	🟨	🟨	🟨	🟨
	Attribute Write StatusCode & Timestamp				🟨
	Attribute Write Index	🟨	🟨	🟨	🟨
	Method Call			🟩	🟩

		Nano	Micro	Embedded	Standard
View Services	View Basic	🟩	🟩	🟩	🟩
	View TranslateBrowsePath	🟩	🟩	🟩	🟩
	View RegisterNodes	🟩	🟩	🟩	🟩
	View Minimum Continuation Point 01	🟩	🟩	🟩	🟩
	View Minimum Continuation Point 05				🟩
Security	Security User Name Password	🟩	🟩	🟩	🟩
	Security User X509				🟩
	Security Invalid user token	🟩	🟩	🟩	🟩
	Security Default ApplicationInstance Certificate			🟩	🟩
	Authentication			🟨	🟨
	Security Policy Required			🟩	🟩
	Security None CreateSession ActivateSession	🟩	🟩	🟩	🟩
	Security Administration	🟨	🟨	🟨	🟨
	SymmetricSignatureAlgorithm_None	🟩	🟩	🟩	🟩
	SymmetricEncryptionAlgorithm_None	🟩	🟩	🟩	🟩
	AsymmetricSignatureAlgorithm_None	🟩	🟩	🟩	🟩
	AsymmetricEncryptionAlgorithm_None	🟩	🟩	🟩	🟩
	KeyDerivationAlgorithm_None	🟩	🟩	🟩	🟩
	SecurityPolicy_None_Limits	🟩	🟩	🟩	🟩
	Security Basic 256	🟨	🟨	🟨	🟨
	Security Basic 256 Sha256	🟨	🟨	🟨	🟨
Caption	Coded in C				
	Coded in B				
	Mandatory feature supported	🟩	🟩	🟩	🟩
	Optional feature not supported	🟨	🟨	🟨	🟨

S2OPC server functional scope

For example:

- Address space base is mandatory from Nano and supported by S2OPC,
- Attribute Write Values is optional from Nano and supported by S2OPC,
- Subscription Publish Discard Policy is mandatory from Micro and is not supported by S2OPC.

ANNEX D S2OPC SERVER DEMO API

S2OPC demo server shall be parameterized using dedicated XML file to:

- Configure server (certificates, endpoints, namespace, etc.),
- Configure users authentication and authorization,
- Define the server OPC UA address space content.

S2OPC demo server shall be configured thanks to an XML file. This includes:

- Endpoint URL
- Security policies
- Certificates

The corresponding schema can be found in S2OPC Gitlab repository in https://gitlab.com/systerel/S2OPC/-/blob/master/schemas/s2opc_clientserver_config.xsd.

S2OPC demo server users authentication and authorization (global read /write/execute access) shall be configured thanks to an XML file. The corresponding schema can be found in S2OPC Gitlab repository in https://gitlab.com/systerel/S2OPC/-/blob/master/schemas/s2opc_clientserver_users_config.xsd.

S2OPC demo server OPC UA address space content shall be configured thanks to an XML file. The corresponding schema can be found in S2OPC Gitlab repository in https://gitlab.com/systerel/S2OPC/-/blob/master/schemas/UANodeSet.xsd.

Example using pre-defined demonstration XML files:

```
$ TEST_SERVER_XML_CONFIG=./S2OPC_Server_CSPN_ST_Config.xml
TEST_USERS_XML_CONFIG=./S2OPC_Users_Demo_Config.xml
TEST_SERVER_XML_ADDRESS_SPACE=./S2OPC_Demo_NodeSet.xml ./toolkit_demo_server
```

Note1: the password to decrypt the server private key needs to be provided interactively in the terminal at startup. The password used to encrypt the demonstration private keys is “password”. New private keys shall be generated and encrypted using a password compliant with ANSSI guidelines (see [R15]).

Note 2: demonstration session users with associated password (stored as PBKDF2 with HMAC-SHA256 hashes) are defined in `./S2OPC_Users_Demo_Config.xml` file. New users and associated passwords shall be configured using a password compliant with ANSSI guidelines (see [R15]). Information on how to generate new password hashes are provided here: https://gitlab.com/systerel/S2OPC/-/wikis/Demo#how-to-use-the-password-pbkdf2-hashes-generation-script.

ANNEX E S2OPC CLIENT DEMO API

S2OPC demo clients are several binaries dedicated to an OPC UA service.

Default configuration of binaries uses security mode “Sign” and security policy “Basic256Sha256”.

Note 1: for each demo, the password to decrypt the client private key needs to be provided interactively in the terminal at startup. The password used to encrypt the demonstration private keys is “password”. New private keys shall be generated and encrypted using a password compliant with ANSSI guidelines (see [R15]).

Note 2: for each demo except discovery, “user1” username is considered to be an allowed user for session activation in server . The demonstration “user1” session user password is “password” in the demonstration configuration of the security target server.

s2opc_discovery

Description	This demonstrator opens a secure channel, sends a GetEndPointsRequest, prints the result then closes the secure channel.

Parameters	
	```
	Usage: s2opc_discovery [options]

	S2OPC discovery demo: get endpoints from a server

	Connection options
	    -e, --endpointURL=<str>   (default: opc.tcp://localhost:4841) endpoint URL in format: opc.tcp://<ip>:<port>[/<name>]
	    --none                   (default: false) use None mode and policy for the connection. Otherwise Sign mode is used with policy Basic256Sha256.
	    --encrypt                (default: false) use SignAndEncrypt (!none required) mode for the connection with policy Basic256Sha256
	    --scLifetime=<int>       (default: 60000 ms) secure channel lifetime (symmetric key renewal)
	    --client_cert=<str>      (default: ./client_public/client_4k_cert.der) path to the client certificate to use (public key)
	    --client_key=<str>       (default: ./client_private/encrypted_client_4k_key.pem) path to the client private key to use
	    --server_cert=<str>      (default: ./server_public/server_4k_cert.der) path to the server certificate to use
	    --ca=<str>               (default: ./trusted/cacert.der) path to the certificate authority (CA)
	    --crl=<str>              (default: ./revoked/cacrl.der) path to the certificate authority revocation list (CRL)
	    --issued=<str>           (default: NULL) path to an issued certificate (e.g.: trusted self-signed server certificate)
	    --no_key_encryption      (default: false) set if the client application private key is not encrypted
	Example: ./s2opc_discovery
	```

Return value	0 if the request is successful. Another value is request fails.

Example:

```
$ ./s2opc_discovery
```

Or with encryption (in addition to signature):

```
$./s2opc_discovery --encrypt
```

s2opc_read

Description	This demonstrator opens a session, sends a read request, prints the result and closes the session.
Parameters	see below
Return value	0 if the request is successful. Another value is request fails.

```
    Usage: s2opc_read [options]

    S2OPC read demo: read a node attribute
        -h, --help                show this help message and exit

    Read options
        -n, --node_id=<str>       node id to read
        -a, --attribute_id=<int>  attribute id to read

    Connection options
        -e, --endpointURL=<str>    (default: opc.tcp://localhost:4841) endpoint URL in format:
    opc.tcp://<ip>:<port>[/<name>]
        --none                     (default: false) use None mode and policy for the connection.
    Otherwise Sign mode is used with policy Basic256Sha256.
        --encrypt                  (default: false) use SignAndEncrypt (!none required) mode for
    the connection with policy Basic256Sha256
        --scLifetime=<int>         (default: 60000 ms) secure channel lifetime (symmetric key
    renewal)
        --client_cert=<str>        (default: ./client_public/client_4k_cert.der) path to the
    client certificate to use (public key)
        --client_key=<str>         (default: ./client_private/encrypted_client_4k_key.pem) path
    to the client private key to use
        --server_cert=<str>        (default: ./server_public/server_4k_cert.der) path to the
    server certificate to use
        --ca=<str>                 (default: ./trusted/cacert.der) path to the certificate
    authority (CA)
        --crl=<str>                (default: ./revoked/cacrl.der) path to the certificate
    authority revocation list (CRL)
        --issued=<str>             (default: NULL) path to an issued certificate (e.g.: trusted
    self-signed server certificate)
        --user_policy_id=<str>     (default: 'user') user policy id used to establish session
        -u, --username=<str>       (if anonymous mode is not active) the username of the user
    used to establish session. If set the user password will be requested in terminal.
        --sessionName=<str>        (default: 'S2OPC_client_session') the session name indicated
    server on session creation
        --no_key_encryption        (default: false) set if the client application private key is
    not encrypted

    Expects at least 2 arguments:
     -n: the Node id XML formatted [ns=<digits>;]<i, s, g or b>=<nodeid>,
     -a: the Attribute id as an int:
                        NodeId |  1
                     NodeClass |  2
                    BrowseName |  3
                   DisplayName |  4
                   Description |  5
                     WriteMask |  6
                 UserWriteMask |  7
                    IsAbstract |  8
                     Symmetric |  9
                   InverseName | 10
               ContainsNoLoops | 11
                 EventNotifier | 12
                         Value | 13
                      DataType | 14
                     ValueRank | 15
               ArrayDimensions | 16
                   AccessLevel | 17
               UserAccessLevel | 18
       MinimumSamplingInterval | 19
                   Historizing | 20
                    Executable | 21
                UserExecutable | 22
E.g.: ./s2opc_read -n i=2259 -a 13
```

Example (if anonymous user have read access to address space):

```
$ ./s2opc_read -u user1 -n i=2259 -a 13
```

Or with encryption (in addition to signature of message content):

```
$ ./s2opc_read -u user1 --encrypt -n i=2259 -a 13
```

s2opc_write

Description	This demonstrator opens a session, sends a write request, prints the result and closes the session.
Parameters	Usage: s2opc_write [options] <value> S2OPC write demo: write a node Value attribute with numerical type -h, --help show this help message and exit Write options -n, --node_id=<str> node id to read -t, --builtin_id=<int> OPC UA built in id type to write Connection options -e, --endpointURL=<str> (default: opc.tcp://localhost:4841) endpoint URL in format: opc.tcp://<ip>:<port>[/<name>] --none (default: false) use None mode and policy for the connection. Otherwise Sign mode is used with policy Basic256Sha256. --encrypt (default: false) use SignAndEncrypt (!none required) mode for the connection with policy Basic256Sha256 --scLifetime=<int> (default: 60000 ms) secure channel lifetime (symmetric key renewal) --client_cert=<str> (default: ./client_public/client_4k_cert.der) path to the client certificate to use (public key) --client_key=<str> (default: ./client_private/encrypted_client_4k_key.pem) path to the client private key to use --server_cert=<str> (default: ./server_public/server_4k_cert.der) path to the server certificate to use --ca=<str> (default: ./trusted/cacert.der) path to the certificate authority (CA) --crl=<str> (default: ./revoked/cacrl.der) path to the certificate authority revocation list (CRL) --issued=<str> (default: NULL) path to an issued certificate (e.g.: trusted self-signed server certificate) --user_policy_id=<str> (default: 'user') user policy id used to establish session -u, --username=<str> (if anonymous mode is not active) the username of the user used to establish session. If set the user password will be requested in terminal. --sessionName=<str> (default: 'S2OPC_client_session') the session name indicated server on session creation --no_key_encryption (default: false) set if the client application private key is not encrypted Expects at least 2 arguments: -n: the Node id XML formatted [ns=<digits>;]<i, s, g or b>=<nodeid>, -t: the BuiltInType id: SOPC_Boolean_Id	1 SOPC_SByte_Id	2 SOPC_Byte_Id	3 SOPC_Int16_Id	4 SOPC_UInt16_Id	5 SOPC_Int32_Id	6 SOPC_UInt32_Id	7 SOPC_Int64_Id	8 SOPC_UInt64_Id	9 SOPC_Float_Id	10 SOPC_Double_Id	11 SOPC_String_Id	12 E.g.: ./s2opc_write -u user1 -n "ns=1;s=Byte_001" -t 3 42
Return value	0 if the request is successful. Another value is request fails.

Example (if “user1” have write access to address space):

```
$ ./s2opc_write -u user1 -n "ns=1;s=Byte_001" -t 3 42
```

Or with encryption (in addition to signature of message content):

```
$ ./s2opc_write --encrypt -u user1 -n "ns=1;s=Byte_001" -t 3 42
```

s2opc_browse

Description	This demonstrator opens a session, sends a browse request, prints the result and closes the session.
Parameters	Usage: s2opc_browse [options] S2OPC browse demo: browse a node in address space -h, --help show this help message and exit Browse options -n, --node_id=<str> node id to browse Connection options -e, --endpointURL=<str> (default: opc.tcp://localhost:4841) endpoint URL in format: opc.tcp://<ip>:<port>[/<name>] --none (default: false) use None mode and policy for the connection. Otherwise Sign mode is used with policy Basic256Sha256. --encrypt (default: false) use SignAndEncrypt (!none required) mode for the connection with policy Basic256Sha256 --scLifetime=<int> (default: 60000 ms) secure channel lifetime (symmetric key renewal) --client_cert=<str> (default: ./client_public/client_4k_cert.der) path to the client certificate to use (public key) --client_key=<str> (default: ./client_private/encrypted_client_4k_key.pem) path to the client private key to use --server_cert=<str> (default: ./server_public/server_4k_cert.der) path to the server certificate to use --ca=<str> (default: ./trusted/cacert.der) path to the certificate authority (CA) --crl=<str> (default: ./revoked/cacrl.der) path to the certificate authority revocation list (CRL) --issued=<str> (default: NULL) path to an issued certificate (e.g.: trusted self-signed server certificate) --user_policy_id=<str> (default: 'user') user policy id used to establish session -u, --username=<str> (if anonymous mode is not active) the username of the user used to establish session. If set the user password will be requested in terminal. --sessionName=<str> (default: 'S2OPC_client_session') the session name indicated server on session creation --no_key_encryption (default: false) set if the client application private key is not encrypted Expects at least 1 argument: -n: the Node id XML formatted [ns=<digits>;]<i, s, g or b>=<nodeid>, E.g.: ./s2opc_browse -n i=85
Return value	0 if the request is successful. Another value is request fails.

Example:

```
$ ./s2opc_browse -u user1 -n i=85
```

Or with encryption (in addition to signature of message content):

```
$ ./s2opc_browse -u user1 --encrypt -n i=85
```

END OF DOCUMENT