PD ISO/IEC TS 30168:2024

BSI Standards Publication

Internet of Things (IoT) — Generic trust anchor application programming interface for industrial IoT devices

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

National foreword

This Published Document is the UK implementation of ISO/IEC TS 30168:2024.

The UK participation in its preparation was entrusted to Technical Committee IOT/1, Internet of Things.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

This publication is not to be regarded as a British Standard.

© The British Standards Institution 2024
Published by BSI Standards Limited 2024

ISBN 978 0 539 30124 3

ICS 35.020

Compliance with a Published Document cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 May 2024.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

ISO/IEC TS 30168

Edition 1.0 2024-05

TECHNICAL SPECIFICATION

colour inside

Internet of Things (IoT) – Generic trust anchor application programming interface for industrial IoT devices

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 35.020

ISBN 978-2-8322-8518-3

Warning! Make sure that you obtained this publication from an authorized distributor.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

CONTENTS

FOREWORD 7
INTRODUCTION 9
1 Scope 10
2 Normative references 10
3 Terms and definitions 10
4 Abbreviated terms 12
5 Architecture 14
5.1 General 14
5.2 Relation to ISO/IEC 30141 14
5.3 Intended target environment 14
5.4 Functional scope 15
5.5 Concepts 15
5.5.1 Abstraction 15
5.5.2 Object information model 20
5.5.3 Identifiers 22
5.5.4 Personalities 23
5.5.5 Profiles 24
5.5.6 Device states 25
5.5.7 Access control 25
5.5.8 Secure element properties 26
5.6 Implementation view 28
5.6.1 System design considerations 28
5.6.2 Personalities 29
5.6.3 Profiles 30
5.6.4 Device states 32
5.6.5 Access control 37
5.6.6 GTA API start-up 40
6 API specification 41
6.1 Overview 41
6.2 Language binding 46
6.3 Endianness 46
6.4 Exception handling 46
6.5 Using GTA API from an application 46
6.5.1 Header files 46
6.5.2 Call conventions and error handling 46
6.6 Types and function documentation 47
6.6.1 Basic types 47
6.6.2 General management functions 50
6.6.3 Process synchronization 55
6.6.4 Secure memory management 59
6.6.5 Function parameter I/O streams 61
6.6.6 Instance management functions 65
6.6.7 Context management functions 67
6.6.8 Access token functions 71
6.6.9 Device state management functions 75
6.6.10 Identifier and personality management 77

AVA Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

6.6.11 Access policy management functions ... 97
6.6.12 Data protection functions ... 106
6.6.13 Channel protection functions ... 109
6.6.14 Supplementary security functions ... 114
6.6.15 Trusted execution environment ... 115
6.6.16 Secure element provider implementation support ... 115
Annex A (normative) GTA API C header files ... 119
A.1 Dependencies ... 119
A.2 Application interface – gta_api.h ... 119
A.3 Provider interface – gta_apif.h ... 119
A.4 Handles – gta_handle.h ... 119
A.5 Function parameter I/O streams – gta_stream.h ... 120
A.6 Error information – gta_errinfo.h ... 120
A.7 Secure memory management – gta_secmen.h ... 120
A.8 Process synchronization – gta_psync.h ... 120
Annex B (normative) Basic profiles ... 121
B.1 ch.iec.30168.basic.passcode ... 121
B.1.1 Description ... 121
B.1.2 Deployment ... 121
B.1.3 Usage ... 122
B.2 ch.iec.30168.basic.local_data_integrity_only ... 122
B.2.1 Description ... 122
B.2.2 Creation ... 122
B.2.3 Usage ... 123
B.3 ch.iec.30168.basic.local_data_protection ... 124
B.3.1 Description ... 124
B.3.2 Creation ... 124
B.3.3 Usage ... 124
Annex C (informative) Example security scenarios for Industrial IoT ... 126
C.1 Analysis of example security scenarios for IIoT ... 126
C.1.1 General ... 126
C.1.2 Scenarios for application protocols ... 126
C.1.3 Secure device identities ... 131
C.1.4 Supply-chain and trustworthiness/authenticity of device ... 132
C.1.5 Device integrity protection ... 133
C.1.6 Application security ... 134
C.1.7 Feature licensing ... 136
C.1.8 Device and machine management ... 137
C.1.9 Blockchain/distributed ledger technology ... 140
C.1.10 GTA management ... 141
C.2 Security requirements for security scenarios ... 142
C.2.1 General ... 142
C.2.2 General or nonfunctional requirements ... 142
C.2.3 Functional security requirements overview and description ... 143
C.2.4 Security requirements for OPC UA ... 145
C.2.5 Security requirements for PROFINET security extensions ... 145
C.2.6 Security requirements for secure communication ... 146
C.2.7 Security requirements for secure device identities ... 146
C.2.8 Security requirements for trustworthiness/authenticity of device ... 147

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

C.2.9 Security requirements for device integrity protection ... 147
C.2.10 Security requirements for application security ... 148
C.2.11 Security requirements for feature licensing ... 148
C.2.12 Security requirements for device management ... 149
C.2.13 Security requirements for blockchain/distributed ledger technology ... 149
C.2.14 Security requirements for GTA management ... 150
Annex D (informative) Security classes and attestation ... 151
D.1 Security classes/levels ... 151
D.2 Offline validation of security level by organizational means (out-of-band) ... 152
D.3 Online validation of security level by attestation (in-band) ... 152
D.3.1 General ... 152
D.3.2 Attestation of SE or GTA API runtime for a specific device ... 152
D.3.3 Attestation of personalities and their attributes ... 152
D.3.4 Attestation of a transaction ... 153
D.3.5 Detached attestation ... 153
Annex E (informative) Examples for further illustration of GTA API concepts ... 154
E.1 Pre-initial device state example for TPM ... 154
E.2 Composing systems from subsystems containing SEs ... 155
E.3 Example deployment of SEs in a composite system design ... 156
Annex F (informative) Implementation guidance ... 160
F.1 Host platform abstraction ... 160
F.2 Buffer management ... 160
F.3 Signalling and semaphores ... 160
F.4 Coding style ... 161
F.5 Secure coding ... 161
Annex G (informative) Example code ... 162
G.1 General ... 162
G.2 Using GTA API with `ch.iec.30168.basic.local_data_protection` ... 162
G.3 Using GTA API with `ch.iec.30168.basic.local_data_integrity_only` ... 165
G.3.1 General ... 165
G.3.2 Protection with data recovery ... 165
G.3.3 Detached protection ... 165
G.4 Protecting a personality with `ch.iec.30168.basic.passcode` ... 167
G.5 Example for a simple buffer stream ... 170
G.5.1 myio_bufstream.h ... 170
G.5.2 myio_bufstream.c ... 172
G.6 Secure element provider template ... 174
Bibliography ... 175

Figure 1 – GTA API environment ... 15
Figure 2 – GTA API modular architecture interfaces ... 16
Figure 3 – Crypto technology driven API design ... 18
Figure 4 – GTA API security service driven API design ... 19
Figure 5 – Multi-application capability ... 19
Figure 6 – Secure element abstraction ... 20
Figure 7 – Object information model (static view) ... 21

AVA Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Figure 8 – Object information model (runtime view) 22
Figure 9 – Value creation chain 25
Figure 10 – Device state stack 33
Figure 11 – Device state transitions 33
Figure 12 – Device state stack (push) 34
Figure 13 – Device state stack (pop) 34
Figure 14 – Access token 37
Figure 15 – Personality derived access token 39
Figure 16 – Access policy composition (BNF) 40
Figure 17 – GTA API start-up phases 41
Figure 18 – Example gta_personality_enumerate() 78
Figure 19 – Example access policy handling by SE provider 99
Figure 20 – Channel protection functions 110
Figure A.1 – Dependency graph for `gta_api.h` 119
Figure C.1 – Device management 138
Figure E.1 – Composing systems from subsystems containing SEs 155
Figure E.2 – Example: Robot as a composite system 156
Figure E.3 – Example: SEs deployed within composite system 157
Figure E.4 – Example: Component device states 159

Table 1 – Access control 26
Table 2 – Mapping between SE properties and protection goals 28
Table 3 – Properties of personality creation profiles 31
Table 4 – Properties of personality deployment profiles 31
Table 5 – Properties of personality enrollment profiles 31
Table 6 – Properties of personality usage profiles 32
Table 7 – GTA API function groups 42
Table 8 – GTA API feature classes 43
Table 9 – GTA API functions per feature class 43
Table 10 – Basic profiles 50
Table 11 – GTA API functions with access control 71
Table B.1 – ch.iec.30168.basic.passcode deployment properties 121
Table B.2 – ch.iec.30168.basic.passcode usage properties 122
Table B.3 – ch.iec.30168.basic.local_data_integrity_only creation properties 123
Table B.4 – ch.iec.30168.basic.local_data_integrity_only usage properties 123
Table B.5 – ch.iec.30168.basic.local_data_protection creation properties 124
Table B.6 – ch.iec.30168.basic.local_data_protection usage properties 125
Table C.1 – Scenarios for OPC UA client and server 127
Table C.2 – Security classes for the PROFINET protocol 130
Table C.3 – Scenarios for PROFINET security 130
Table C.4 – Scenarios for secure communication protocols 131
Table C.5 – Scenarios for secure identities 132
Table C.6 – Scenarios for device trustworthiness 133

AVA Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Table C.7 – Scenarios for system integrity protection 134
Table C.8 – Scenarios for know-how protection 135
Table C.9 – Scenarios for feature licensing 137
Table C.10 – Scenarios for device management 139
Table C.11 – Scenarios for blockchain/distributed ledger technology (DLT) 141
Table C.12 – Scenarios for GTA management 142
Table C.13 – General or nonfunctional requirements 142
Table C.14 – GTA-API functional security requirements 143
Table C.15 – Security requirements for OPC UA 145
Table C.16 – Security requirements for PROFINET security extensions 146
Table C.17 – Security requirements for secure communication 146
Table C.18 – Security requirements for secure device identities 147
Table C.19 – Security requirements for trustworthiness/authenticity of device 147
Table C.20 – Security requirements for device integrity protection 148
Table C.21 – Security requirements for application security 148
Table C.22 – Security requirements for feature licensing 149
Table C.23 – Security requirements for device management 149
Table C.24 – Security requirements for blockchain/distributed ledger technology 150
Table C.25 – Security requirements for GTA management 150
Table D.1 – Example security levels 151

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

INTERNET OF THINGS (IoT) – GENERIC TRUST ANCHOR APPLICATION PROGRAMMING INTERFACE FOR INDUSTRIAL IoT DEVICES

FOREWORD

1) ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

2) The formal decisions or agreements of IEC and ISO 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 and ISO National bodies.

3) IEC and ISO documents have the form of recommendations for international use and are accepted by IEC and ISO National bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC and ISO documents is accurate, IEC and ISO 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 and ISO National bodies undertake to apply IEC and ISO documents transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC and ISO document and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and ISO do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC and ISO marks of conformity. IEC and ISO are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this document.

7) No liability shall attach to IEC and ISO or their directors, employees, servants or agents including individual experts and members of its technical committees and IEC and ISO National bodies 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 ISO/IEC document or any other IEC and ISO documents.

8) Attention is drawn to the Normative references cited in this document. Use of the referenced publications is indispensable for the correct application of this document.

9) IEC and ISO draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC and ISO take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC and ISO had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at https://patents.iec.ch and www.iso.org/patents. IEC and ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TS 30168 has been prepared by subcommittee 41: Internet of Things and Digital Twin, of IEC technical committee JTC 1: Information technology. It is a Technical Specification.

This document contains attached files in the form of GTA API C header files and a secure element provider template that are cited in Annex A and Clause G.6. These files are intended to be used as a complement and do not form an integral part of the publication.

The text of this Technical Specification is based on the following documents:

Draft	Report on voting
JTC1-SC41/388/DTS	JTC1-SC41/413/RVDTS

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

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

The language used for the development of this Technical Specification is English.

IMPORTANT – The "colour inside" logo on the cover page of this document 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.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

INTRODUCTION

Industrial Internet of Things (IIoT) devices face increasing security requirements. This insight is especially important as more and more devices become connected directly or indirectly to the Internet. It is essential that IIoT devices are prepared to perform secure communications for service interaction, monitoring, and control.

However, security is often still observed as rather complex by implementers and integrators. This perception often results in realization obstacles when the integration and use of security mechanisms and secure elements (SE) is wanted.

This document provides a versatile application programming interface (API) for security to allow a generic integration of SEs into IIoT devices. The API is vendor independent and also independent regarding the SE technology being deployed. This approach simplifies redesign for different SEs and supports software-hardware co-design for security. SEs offering different security properties facilitate the selection of an SE according to the intended use, protection goals, and other boundary conditions. The API aims at achieving high-level abstraction profiles for security services and mechanisms to avoid typical low-level interoperability complexity and implementation failures. Requirements and architectural constraints from IIoT applications shape the final design of the API and its usability.

The resulting API facilitates the security-by-design defined integration of security components within IIoT components on a large scale. The time-to-market for secured devices is accelerated. Stakeholders will benefit from higher security levels being available for lower prices. Application of updates and continuous improvements of security along the lifecycle of products and systems are facilitated.

The following stakeholders and their corresponding interests play a role for the generic trust anchor application programming interface (GTA API) definition:

- Manufacturers and users of industrial equipment

  Scalable use of adequate (hardware-based) security technologies depending on required security, multivendor support, migration strategy, or long-term suitability.
- Software developers

  Increased robustness due to use of a unified API.
  Ease of use for developers without dedicated security expertise.
- Manufacturers of security ICs or ICs offering security functions

  Promote use of hardware-based trust anchor technologies for IIoT devices.
- Conformity Assessment Bodies

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

INTERNET OF THINGS (IoT) – GENERIC TRUST ANCHOR APPLICATION PROGRAMMING INTERFACE FOR INDUSTRIAL IoT DEVICES

1 Scope

This document specifies a generic application programming interface (API) for the integration of SEs within Industrial Internet of Things (IIoT) devices. It considers needs from industrial usage scenarios and applications. This document also provides guidance for implementation, testing, and conformity validation.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEEE Std 802-2014, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*

IETF RFC 1035, P. Mockapetris, "Domain names – implementation and specification", November 1987, available at https://www.rfc-editor.org/info/rfc1035 [viewed 2023-08-29]

IETF RFC 1779, S. Kille, "A String Representation of Distinguished Names", March 1995, available at https://www.rfc-editor.org/info/rfc1779 [viewed 2023-08-29]

IETF RFC 4122, P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", July 2005, available at https://www.rfc-editor.org/info/rfc4122 [viewed 2023-08-29]

IETF RFC 8446, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", August 2018, available at https://www.rfc-editor.org/info/rfc8446 [viewed 2023-08-29]

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- IEC Electropedia: available at https://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp

3.1
attack
attempt to gain access to an information processing system in order to produce damage

Note 1 to entry: The damage can be, for example, destruction, disclosure, alteration, unauthorized use.

[SOURCE: IEC 60050-171:2019, 171-08-12]

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

3.2
generic trust anchor application programming interface
GTA API
set of well-defined methods, functions, routines, or commands for application software to facilitate the programming languages use of cryptographic or protected resources from an SE that is used as trust anchor

3.3
IoT device
entity of an IoT system that interacts and communicates with the physical world through sensing or actuating

Note 1 to entry: Industrial IoT device is an IoT device which is intended for use by qualified and experience personnel in a controlled manufacturing or processing environment.

Note 2 to entry: For example, industrial uses are smart manufacturing, robot, energy, automobile, transportation, building, and so on. Such an entity can interact with digital twins or asset administration shells of other entities.

Note 3 to entry: An entity can be the combination of the device and a simple sensor. The device comes with a GTA API instance. The sensor does not have an intelligent control unit and does not come with its own GTA API instance.

Note 4 to entry: Industrial IoT devices are not limited to sensor or actuator. Industrial IoT devices include embedded devices such as smart sensor/actuator, programmable logic controller (PLC), edge device (for example, Industrial PC) but not limited to sensor and actuator.

[SOURCE: IEC 60050-741:2020, 741-02-04, modified – The original Note to entry has been replaced with Notes explaining the use of the term in an industrial context.]

3.4
know-how protection
measures supporting the protection against extraction of knowledge or expertise represented by software or other intellectual property from a device

3.5
personality
set of trusted information and cryptographic key material that is used by an application in a specific security context

Note 1 to entry: A personality typically includes the device's own cryptographic material like private keys, secret keys, own public key certificates. A personality can also include information required to establish trust towards partners.

Note 2 to entry: A personality is used within the scope of a specific application. A device can use different personalities in different application contexts.

3.6
public key certificate
set of data that uniquely identifies an entity, contains the public key of the entity, and is digitally signed by a trusted party to bind the public key to the entity

3.7
secure element
SE
component capable of securely hosting functionalities or confidential and cryptographic data or both in accordance with well-defined rules and security requirements

Note 1 to entry: A typical solution for an SE is a one chip microcontroller.

Note 2 to entry: Cryptographic keys are an example of confidential and cryptographic data.

Note 3 to entry: An SE can be realized as pure software component to support future migration to a hardware SE.

Note 4 to entry: An SE can provide special physical protection features such as tamper protection.

AVA Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

3.8
side-channel analysis
exploitation of the fact that the instantaneous side-channels emitted by a cryptographic device depends on the data it processes and on the operation it performs to retrieve secret parameters

[SOURCE: ISO/IEC 17825:2024, 3.9]

3.9
trust anchor
essential security capability that, by definition, must be trusted

Note 1 to entry: A trust anchor can provide provisions to protect the integrity and confidentiality of functions and related information that are required by an application.

Note 2 to entry: The security capability to achieve protection can be provided with an SE. The SE can provide functionality for, for example, secure generation and use of cryptographic key material, and tamper-protected storage of public key certificates.

Note 3 to entry: Data sets, for example, public key certificates starting a certification path, require additional protection against manipulation to be considered as trust anchor. This additional protection can be achieved by, for example, storage in a shielded location.

4 Abbreviated terms

AES	Advanced Encryption Standard
ASN.1	Abstract Syntax Notation One
CA	Certification Authority
CERT	Computer Emergency Response Team
CHAP	Challenge handshake authentication protocol
CO	Component
DCP	Discovery and basic Configuration Protocol
DER	Distinguished Encoding Rules
DH	Diffie-Hellman
DNS	Domain Name System
DSS	Digital Signature Scheme
DTLS	Datagram Transport Layer Security
EC	Elliptic Curve
EK	Endorsement Key
ECDSA	Elliptic Curve Digital Signature Algorithm
FGPA	Field Programmable Gate Array
GDS	Generic Discovery Service
HSM	Hardware Security Module
IACS	Industrial automation and control system
IANA	Internet Assigned Numbers Authority
IDevID	Initial Device Identifier
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPC	Industrial Personal Computer
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

IIoT	Industrial IoT
IoT	Internet of Things
IT	Information Technology
LDevID	Local Device Identifier
MAC	Media Access Control
M2M	Machine to machine
NIC	Network Interface Card
NV	Nonvolatile
NVRAM	Nonvolatile Random Access Memory
NVindex	Nonvolatile Index
OPC	Open Platform Communications
OPC UA	Open Platforms Communications Unified Architecture
OS	Operating System
PAKE	Password-authenticated key exchange
PC	Personal Computer
PCR	Platform Control Register
PG	Protection Goal
PI	PROFINET International
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
PKIX	Public Key Infrastructure X.509
PLC	Programmable Logic Controller
PUF	Physical Unclonable Function
RSA	Rivest Shamir Adleman
SC	Security Controller
SE	Secure Element
SEI	Software Engineering Institute
SEP	Secure Element Properties
SHA	Secure Hash Algorithm
SNMP	Simple Network Management Protocol
SoC	System on chip
SSH	Secure Shell
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOFU	Trust on First Use
TPM	Trusted Platform Module
TSS	TPM Software Stack
UCS	Universal Coded Character Set
UTF-8	UCS Transformation Format

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

5 Architecture

5.1 General

The generic trust anchor application programming interface (GTA API) has been developed to facilitate and implement patterns concerning the security architecture of IIoT devices:

- Trust management process along the lifecycle of IIoT devices (5.5.6)
- Separation of security domains between multiple IIoT applications on a single IIoT device (5.5.1.3, 5.5.7)
- Integration of SEs into the overall security architecture of an IIoT device (5.5.7, 5.6.6)

One of the main objectives of the GTA API is to offload most of the security critical implementation details from the IIoT application developer. The required functionality is encapsulated into a function block that can be maintained independent from the application by security experts.

Annex C provides an in-depth discussion of usage scenarios that are used as a basis for the discussion of GTA API.

Annex E provides some additional examples for the concepts that are introduced in Clause 5.

5.2 Relation to ISO/IEC 30141

The GTA API considers the implementation viewpoint as described by ISO/IEC 30141 [16]. The GTA API provides a high-level application programming interface for the modular integration of SEs into IIoT applications. SEs allow the trustworthy storage and usage of secret and trusted attributes of personalities that are required by IIoT devices to securely interact within an IIoT system. GTA API directly supports the security aspect of the trustworthiness principles for IoT systems that are described in ISO/IEC TS 30149 [17].

5.3 Intended target environment

Architectural decisions for the GTA API are driven by the properties of the intended IIoT and smart manufacturing target environment:

- Embedded devices (such as smart sensors, smart actuators, programmable logic controllers (PLC), edge devices, IIoT devices) with potentially constrained resources
- Unattended use of the device
- Machine to machine (M2M) communication and networked control systems
- "Realtime" operation environments (along with requirements such as deterministic runtime and low latency)

  Realtime is considered to the extent that API design should not inhibit the use of the API under realtime conditions. That means GTA API should support re-entrance and tolerate forced pre-emption by the operating system. It is not assumed that an SE itself meets any special realtime requirements.
- Offline/island mode operation, continuous access to security services such as Public Key Infrastructures is not available, access to those services can be limited to dedicated phases
- Dependable systems in continuous operation mode
- Long-term deployment, systems with long lifecycles

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

GTA API target environments are characterized by engineered components. Before an IIoT device is put into operation, the device manufacturer, the system integrator, or device operator configures how SEs are to be deployed. The following aspects are typically engineered upfront, for example:

- Secure elements that are in place
- Personalities (including secret or trusted attributes) that are required/used

GTA API especially considers the usage in resource constrained devices in industrial automation and control systems (IACS). Devices in an IIoT environment also include dedicated servers within IACSs.

5.4 Functional scope

Provided functions cover the entire lifecycle of a device, starting from initial device manufacture to its final decommissioning (cf. 5.5.6). Over the lifecycle, functionality to configure, engineer, and manage the security on the device is also needed.

Based on the requirements (cf. Clause C.2) the following functions are addressed:

- Secure Storage for secret and trusted attributes of personalities
- Personality Management and Provisioning
- Entity Authentication
- Key Management for Channel Security
- Data Storage Protection
- SE management

In addition to the nonfunctional requirements listed in Clause C.2 within Table C.13, the following is also addressed:

- Platform independency
- Focus on security design (integration issues like resource arbitration or concurrency are better addressed by platform experts)
- Avoidance of unintentional side-effects ("function f does only work if…")
- Avoidance of functions or properties inhibiting Common Criteria, FIPS-140 certification and the like

5.5 Concepts

5.5.1 Abstraction

5.5.1.1 Architectural abstraction

GTA API provides a modular and scalable architecture.

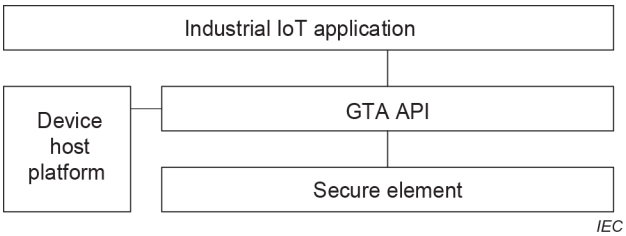

Figure 1 – GTA API environment

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

The links that are shown in Figure 1 indicate the interfaces exposed by the GTA API framework, which are described within this document:

- Accessing GTA API from the IIoT application
- Embedding GTA API into the host platform
- Adapting towards different types of SE, potentially using technology-specific interfaces, for example, ISO/IEC 7816, Trusted Computing Group TPM Software Stack 2.0 (TCG TSS 2.0)

The focus of this document is on the usability at the application level. The internal API design is mostly left to the implementor of the API. However, this document also provides some guidance on how to build a framework that allows functional extension of the API. Using the API to enable cooperation of different SEs is also addressed.

Figure 2 shows the modular architecture and functional segmentation of the GTA API interface. The individual interfaces shown in Figure 2 constitute the GTA API framework. The function interfaces are detailed in Clause 6.

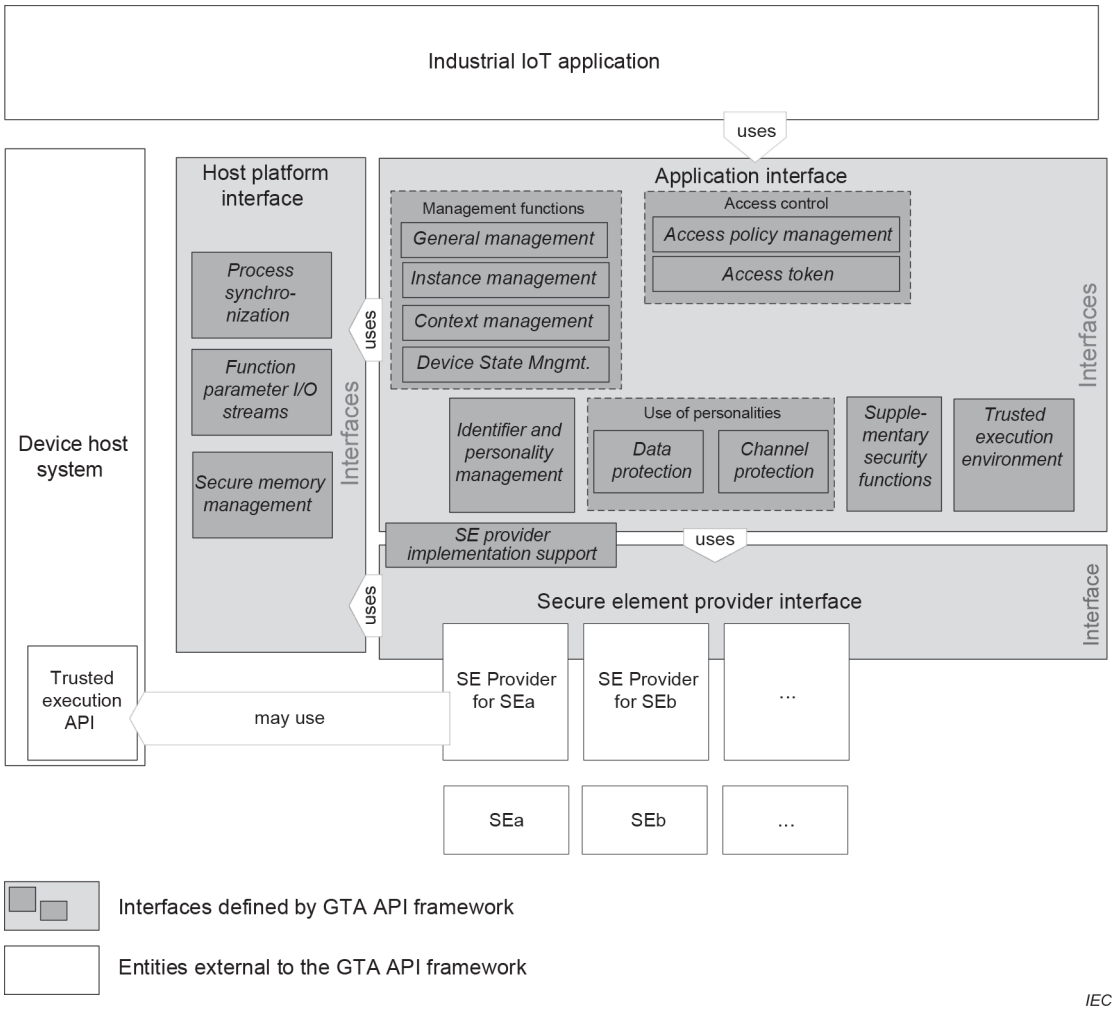

Figure 2 – GTA API modular architecture interfaces

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

The application interface itself is composed of the following main function groups:

- Management functions to manage the GTA API and used SEs.
- Management of the personalities used to express the identity of a device in a specific trust or security domain. The personality of a device is defined by the secret attributes under the control of the device and additional (trusted) attributes, that is, trustworthy information about other devices. Secret attributes can be, for example, cryptographic secrets, private keys, or passwords. Trusted attributes can comprise, for example, cryptographic verifiers, public keys, or certificates.
- Usage of these personalities to protect data or to establish secure associations towards other IIoT devices.
- Supplementary cryptographic functions which are independent of personalities, for example, random number generation.
- Access control for the resources and functionality offered by GTA API.

The GTA API framework allows transparent integration of multiple SEs into an application. The application developer shall be presented with a unified view of all personalities that are provided by all available SEs. SEs using different technologies and being provided by different vendors are attached by SE providers.

From a conceptual point of view, the API fully abstracts the topology that is used to execute API calls. A functional profile that is used in an API call can require execution within an SE that offers some specific characteristic. Such topological aspects can thus be covered by the profile concept (cf. 5.5.5).

Secure elements are attached using SE providers. These providers implement the functionality required to realize a GTA API function according to a selected profile on a specific SE technology. Depending on the SE technology used, the amount of implementation being either software (that is, as part of the provider) or hardware (for example, encapsulated within a physically distinct SE) can vary (cf. 5.6.1.1). The required security level and existing security constraints depend on the intended execution environment. An application running inside a specific execution environment can desire that specific security properties are provided as native SE functions (cf. 5.5.8). Existing technology-specific APIs can be used to facilitate the implementation of SE providers. Examples are the Trusted Platform Module Library Specification [23] or the ISO/IEC 23465 series on cards and security devices for personal identification [24], [25], [26].

SE providers shall be registered within the GTA API framework before the respective SEs can be used by an application. At the time of registration each provider provides a list of callback functions. These callback functions are used by the framework to dispatch and forwards calls from the applications towards the individual providers. The interface between GTA API framework and provider is defined by the subset of GTA API functions potentially depending on functionality that is provided by the SE. The functional abstraction level (cf. 5.5.1.1) at the GTA API application interface and at the provider interface is the same. While the GTA API framework facilitates interfacing with the host system and contains several functions to maintain auxiliary data structures, the framework does not interfere with functions forwarded to the SE provider. The function set defining the provider interface has been moved to a separate header file (cf. Clause A.3 Provider interface – gta_apif.h).

5.5.1.2 Functional abstraction

GTA API is designed to provide a high-level abstraction interface to cryptographic services. Its functions and interfaces follow a top-down, application driven approach.

Figure 3 shows a typical example for a low-level SE technology-specific crypto API.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

```
/*
 * discover public key certifictes and key material
 */

x509_cert = secure_element_read_certificate(
subject_name="/CN=test")

/* ... find matching private key ... */

key_handle = secure_element_get_key(...)

/*
 * setup and perform cryptographic operation
 */

/* ... decide on signature algorithm
 *   based on available key ...
 */

digest = secure_element_hash(
     data = "Hello world!",
     alg = SHA256)

signature = secure_element_sign(
     key_handle,
     digest,
     alg = RSA_PSS)

/* ... do post processing on raw signature ... */
```

Figure 3 – Crypto technology driven API design

This approach provides maximum control over the features and capabilities that are offered by an individual SE. However, usage of such an API requires expert knowledge and creates dependencies towards specific mechanisms and technologies. For example, a signature is not always the best case for data authentication depending on the scenario. Future adaption (for example, use of new algorithms) is often hard to accomplish and costly. This insight is especially true as the lifetime of an industrial application and security mechanisms that are considered to be state-of-the-art is likely to deviate dramatically. It is desirable to be able to exchange substantial parts of the security implementation without an impact to the application.

GTA API aims to hide these implementation details from the application developer. The application developer is able to concentrate on the application. The security experts are able to provide appropriate security according to the intended protection goal (for example, data authentication). Protection goals are addressed by profiles – as discussed in 5.5.5, in a defined trust domain setting – indicated by the personality as discussed in 5.5.4.

Figure 4 shows an example of how functionality like that shown in Figure 3 is expressed using the GTA API.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

```
/* open a GTA context to use a personality with a specific
  profile */

h_ctx = gta_context_open(
      personality = "application1",
      profile = "ch.iec.30168.authenticate_basic")


/* compute an authentication tag according to the selected
  profile */
tag = gta_authenticate_data_detached(
      h_ctx,
      data = "Hello world!")
```

Figure 4 – GTA API security service driven API design

Such high-level service calls as exposed by the GTA API are characterized by certain properties like:

- Only the protection goal is specified.
- Detailed information is contained in an application-specific profile.
- Separation is achieved between application logic and cryptographic knowledge.
- Profile configuration can be updated independent of the application.

Application profiles can be specified by experts that combine deep application domain knowledge with security expertise. This expertise is made available to all application developers or integrators through the GTA API.

5.5.1.3 Multi-application capability

Figure 5 shows a scenario where multiple applications use a single SE in parallel. GTA API shall support environments where multiple applications use resources of the trust anchor in parallel. If so, GTA API shall support the separation of access to the SE resources between these applications. GTA API provides functionality to enable the cooperation among GTA API middleware, SE, and the operating system (OS) platform (5.6.5.2, 5.6.6). This claim means access tokens that are acquired by one application are not valid for another application. Also, any content that is processed by one application is not visible to other applications. In case one personality (cf. 5.5.4) is used by different applications, this multiple use shall be properly orchestrated. The separation is transparent for the applications. That means, the different applications are not expected to be aware of the fact that other applications exist and are accessing the SE in parallel.

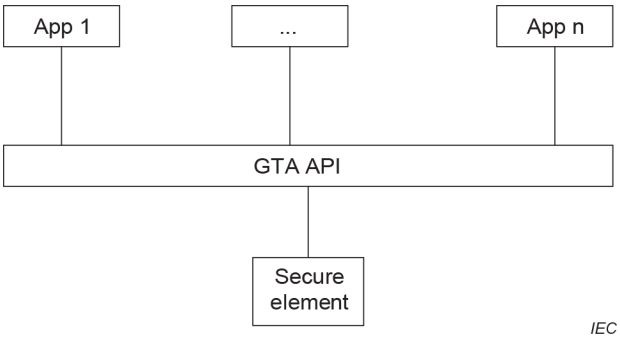

Figure 5 – Multi-application capability

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

5.5.1.4 Deploying multiple SEs in a device

Sometimes multiple SEs are integrated into a component. The integration of SEs from different vendors into the GTA API framework is supported by dedicated SE providers (cf. Figure 6).

There are different reasons for the use of multiple SEs.

- Cooperation of multiple SE types to accomplish a single security function, for example:
  - An SE enabling secure boot to unlock a separate SE, for example, a software SE implementation using an encrypted software key store
  - An SE providing secure key establishment to make session keys available to a high-bandwidth batch encryption device, for example, an Ethernet network interface card (NIC) with integrated symmetric encryption

    To enable cooperation of multiple SEs, exchange of information between several SEs within the GTA API implementation can be required. This kind of information sharing can be achieved by providing a single SE provider encapsulating the functionality of multiple SEs. Alternatively, the exchange of information between multiple SE providers can be enabled.
- Pooling of multiple SEs of the same type. This SE pool can be beneficial in case a single SE is not able to meet the required performance criteria. From the point of view of functional access, the pool of SEs should act in the same way as a single SE. The internal management of the multiple SEs within the pool is vendor specific and shall be encapsulated inside the SE provider.

Regardless of the topologies of SEs installed in a device, the view that is exposed by the GTA API towards the application programmer shall be uniform. That means, the internal organization is transparent to the application programmer.

This abstraction can be addressed by additional API integration layers, like those sketched in Figure 6.

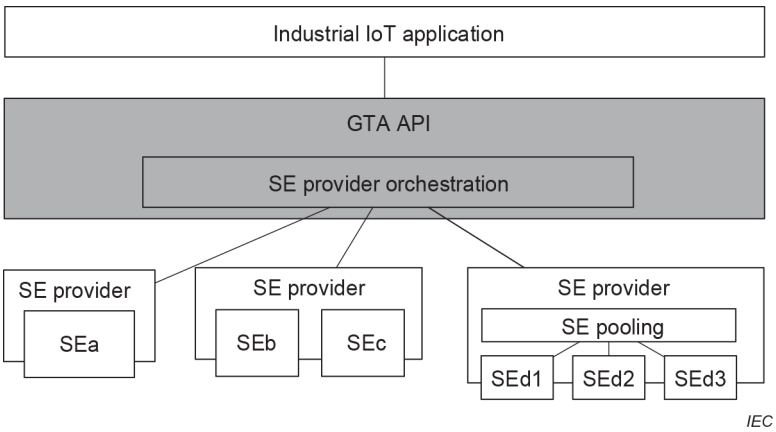

Figure 6 – Secure element abstraction

5.5.2 Object information model

Figure 7 shows the overall object and information model that is used by this document.

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

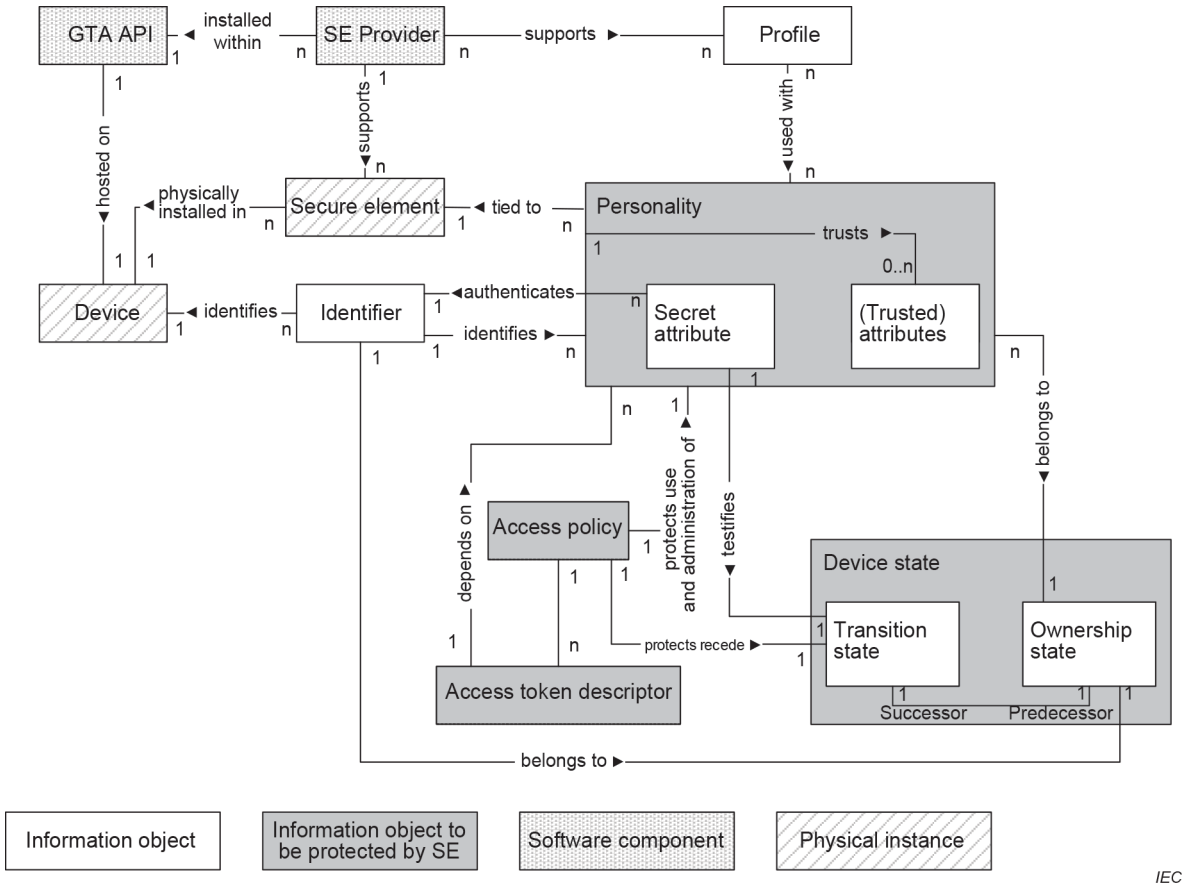

Figure 7 – Object information model (static view)

GTA API is hosted on a device. There is one installation of GTA API for each device.

GTA API may support multiple SEs that are installed in a device. This concept is discussed in detail in 5.5.1.4.

GTA API includes multiple SE providers. Each SE provider allows interaction with a specific type of SE using GTA API. An SE type either refers to a single SE product or a family of SEs.

An SE provider supports different application profiles. The implementor of an SE provider decides whether a specific profile is supported or not. Support for specific profiles also depends on the capabilities of the underlying SE. There can also be distinct SE providers to support multiple profiles for a single type of SE. Further details on profiles are described in 5.5.5.

Various identifiers designated to the device are used to organize and discover personalities. Device identifiers are described in 5.5.3.

Personalities represent the device identity towards the outside world. A personality can act in one of two roles, either as relying party or as requestor. The requestor role uses the secret attributes of the personality to prove the identity of the device. Acting as a relying party a personality relies on trusted attributes to establish trust into a third party. A personality can come with the relying party role only, requestor role only, or act in both roles. Secret attributes are created implicitly during the creation of a new personality and cannot be changed during the lifetime of a personality. Trusted attributes can be dynamically administrated (add/remove/update) during the lifetime of a personality after proper authentication.

Personalities and their attributes are described in 5.5.4 and 5.6.2.1.

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

Device states are used to control management functions of GTA API, especially for the management of personalities. Each personality belongs to a device ownership state. In case a device state is removed, this operation also removes all personalities belonging to that device state. Device states are described in 5.5.6.

Access policies are used to control the usage of personalities and transition state objects. The access policy describes which conditions are to be met to perform specific operations (cf. Table 1) on the respective object. Access policies are described in 5.6.5.3.

The SE is able to provide the required protection for security relevant information objects. The protection can be achieved by either native mechanisms that are provided by the SE (see also 5.5.8) or by hybrid mechanism. A hybrid mechanism can, for example, use checksums that are maintained on the SE.

Figure 8 shows some additional relations applying to GTA API during runtime. Each application uses its own GTA API instance. The current device state is shared by all instances, that is, in case the device state is changed by an application this change affects all applications. New personalities are associated to the current device state. New personalities can become visible to all GTA API instances immediately after creation or may become visible only after an application opened a new GTA API instance.

Contexts are used to organize operations involving a personality. A context provides an environment for GTA API operations. An operation may extend over multiple individual steps each step being an individual GTA API function call, for example, an enrollment process, establishment of a security association. Contexts can also be used to setup preconditions for operations, for example, setting required attributes or authentication conditions.

Authentication conditions are fulfilled by an application by providing the required access tokens as specified by the respective access policies (see above).

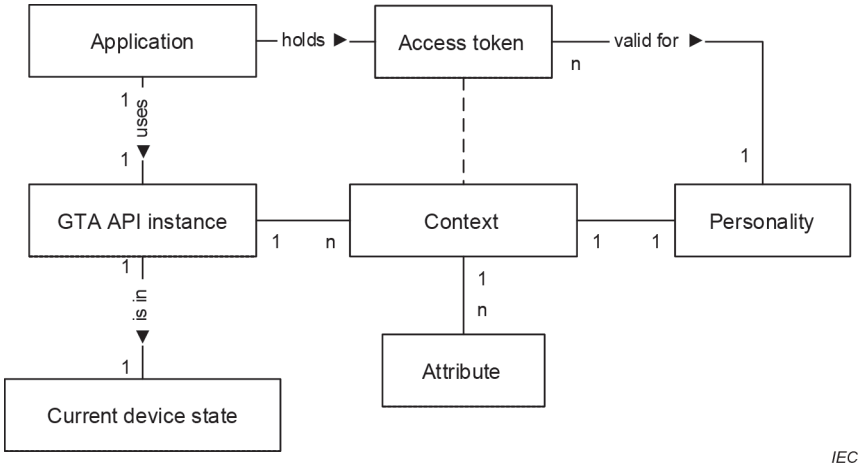

Figure 8 – Object information model (runtime view)

5.5.3 Identifiers

The main purpose of an identifier is to provide a long-term qualifier for the device. This identifier can then be used for comfortable discovery of the device and reference to the device.

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

There are various identifiers that can be used to identify a device depending on the application, for example:

- A domain name system (DNS) name, Internet Protocol (IPv4 or IPv6) address assigned to the device
- A media access control (MAC) address imprinted into the devices network interface card
- A serial number that is assigned to the device during production
- An immutable hardware serial number that is bound to the device or SE during the manufacturing process

GTA API distinguishes these various identifiers by their identifier type. Identifier types shall be constructed using reverse domain name notation, that is, identifier types start with a registered domain name in reverse order[1], for example, `ch.iec.30168.identifier.dns_name`. This notation allows any entity or organization holding a registered domain name to specify unique identifier types.

The identifier as such does not provide any security assurance. To construct a valid proof of the identity of a device, the identifier needs

a) to be bound to the hardware of the device by some means, and
b) to be mapped to a (usually) cryptographic secret attribute that can be used to compute a verifiable artifact on the identity of the device.

To verify the produced artifact, the relying party needs an authentic assertion about the mapping between identifier and the related secret or trusted attribute. The value of the identifier itself should not depend on the value of related secret or trusted attributes. While these attributes eventually change over time (for example, if a key update occurs), the identifier should persist.

The hardware binding is realized by the properties of the SE. Both the computation and the verification of cryptographic artifacts belonging to an identifier are addressed by the concept of personalities.

An SE or device vendor can provide a set of preassigned, static identifiers which are unalterably linked to the device (for example, a serial number). In case such static identifiers are present, one of them should be of type `ch.iec.30168.identifier.se_generic_hw_immutable`.

5.5.4 Personalities

Personalities represent the device towards the outside world. GTA API shall support an IIoT device to act with different personalities. Personalities can occur in the role of a requestor or relying party.

EXAMPLE Downstream an edge device acts as OPC UA client towards a set of resource restricted devices like sensors and actuators to collect and aggregate data close to the edge device. Edge device and the resource restricted devices use one personality inside this context to form a closed community (security domain). The edge device also communicates upstream as OPC UA server with a datacentre which in turn collects analysed and aggregated data from several edge devices. Here the datacentre is the authority to provide and renew personalities for that closed community (security domain). In general, the aim is to separate security domains by using different sets of cryptographic secret or trusted attributes. Separation of distinct security domains is best practice for risk reduction and separation of responsibilities.

Application programmers can combine personalities with profiles to benefit from these security concepts without having to understand the cryptographic details.

Implementation view specific aspects of personalities are discussed in 5.6.2.

[1] See https://en.wikipedia.org/wiki/Reverse_domain_name_notation

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

5.5.5 Profiles

This document uses functional profiles (that is, mandatory functionality of the GTA API instance and its environment) to enable specific scenarios occurring in an IIoT environment.

Instead of specifying cryptography and SE specific individual parameters, the application programmer can specify the respective scenario profile that is required for a specific function. The GTA API ensures that these profiles are realized in an interoperable manner between different implementations. Detailed information – such as algorithms used, key sizes, encapsulation formats – is encapsulated and hidden by the profiles.

Examples for functional scenario profiles are:

- Binary image and data measurement, verification, and authentication to support device and application integrity
- OPC UA authenticated communication
- PROFINET authenticated communication
- Secure communication for substation automation systems

A profile identifies a distinct usage scenario, for example, OPC UA. Profiles encapsulate the complexity of the cryptographic base operations and hide them from the application developer. For example, to compute an authentication signature, instead of selecting a specific cryptographic signature algorithm, digesting scheme, padding, and encapsulation scheme, the application developer straightforwardly selects a profile. The profile was agreed upon by domain and security experts to be suitable for the intended tasks.

Profiles allow the application programmer to benefit from the expertise of security experts. Application programmers are not expected to understand and deal with the details of different types of cryptographic mechanisms. If both peers use the same profile, conformance to the profile ensures the interoperability among peers.

Profiles are identified by profile names as described in 5.6.3.1. Definitions of a profile should contain a clear definition of the intended use case to help the application programmer to select the correct profile. The scenarios described in Clause C.1 can be used as a reference.

Several basic profiles are defined as part of this document within Annex B. These basic profiles are intended to support functions within the GTA API framework (for example, access control). The basic profiles also facilitate some basic use cases (for example, device local data protection).

The basic profiles that are given in Annex B can also be used as examples for the definition of additional profiles. The following aspects should be considered when defining profiles:

- Functional abstraction level: It is possible to define profiles on different abstraction levels. Profiles can be defined on a high level as, for example, the profile ch.iec.30168.basic.local_data_integrity_only given in Clause B.2. The profile ch.iec.30168.basic.local_data_integrity_only leaves a lot of freedom for the provider implementation if the semantic intention is maintained. In contrast, there can be narrowly defined profiles like ch.iec.30168.basic.passcode as given in Clause B.1.
- Interoperability: The profile should take into account the level of interoperability that is required between different implementations for the same profile. This is important in case cryptographic artifacts (for example, protected data) are to be exchanged between devices or other external parties using different implementations. In contrast to this, ch.iec.30168.basic.local_data_integrity_only (see Clause B.2) is an example for a profile which does not require any interoperability among implementations.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

- Minimum functionality: GTA API should provide the minimum functionality that is required regarding the encapsulation of SE-specific functionality. GTA API profiles should not be designed with extensive handling of container formats and cryptographic protocols. This handling is expected to be done by the application or a format or protocol-specific SDK using GTA API to resort to dedicated trust anchor functionality.
- Closure: A profile should include a well-defined functionality. The delivered artifacts are ideally already defined by other standards. Wherever possible, it should be avoided that GTA API and other code are required to manipulate the same artifact, although this is often not completely achievable for legacy protocols.

Implementation view specific aspects of profiles are discussed in 5.6.3.

Clause G.2 provides an example for the usage of a basic profile.

5.5.6 Device states

Device states are used to manage objects such as identifiers and personalities that are created by different device owners. Typically, the device state changes as the device moves from one owner to another along the value creation chain. Examples are the transition from the initial device manufacturer to a machine builder integrating the device into a more complex machine, and finally to the operator deploying the machine into the shop floor. Figure 9 illustrates this value creation chain. Stakeholders and related device states that are often considered within IIoT are used as an example.

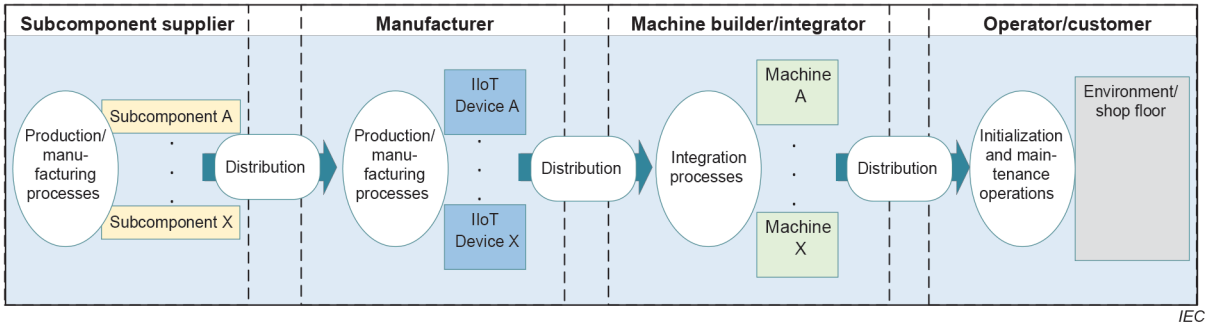

Figure 9 – Value creation chain

Each owner wants the possibility to create its own set of objects on the SE. However, it is important to ensure that subsequent owners are not able to tamper with the settings already installed. It shall also be possible to reset a device to a well-defined state, for example, in case it is moved to a new operational environment.

Implementation view specific aspects of device states are discussed in 5.6.4.

5.5.7 Access control

The execution of several operations on specific objects that are performed through GTA API on an SE are subject to access control. Table 1 lists the objects and the applicable usage restrictions.

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

Table 1 – Access control

Object	Usage	Description
Personality (5.5.4)	Use	Usage of the personality involving a cryptographic computation with a secret attribute
	Admin	Modification of a personality (activation/deactivation) or a trusted attribute (addition/removal), for example, a CA public key certificate, of the personality
Transition device state (5.5.6)	Recede	Recede from a device state to a previous device state; Usage control is determined by the device state to be removed.

Usage restrictions for an object are defined with access policies at the time of object creation and cannot be changed afterwards.

Before using an object that is protected by an access policy, the access policy shall be satisfied by either

- meeting an implicit access condition, or
- providing an access token.

Access conditions depend on the state of GTA API and its runtime environment. Whether an access condition is met or not cannot be influenced by the application attempting to use an object.

Access tokens must be actively acquired and provided by an application attempting to use an object.

Implementation view specific aspects of access control are discussed in 5.6.5.

5.5.8 Secure element properties

This subclause only describes possible properties for SEs that can be used in the context of the GTA API. It does not mandate any of these properties. The intention is to describe specific SE properties (SEPs). These SEPs can be used within other clauses or subclauses of this document or within other documents referencing the GTA API. As an example, they could be used to specify security levels for GTA API specifics like personalities. SEPs can also be used in the context of security classes and attestation (see Annex D). From a GTA API point of view, it is intended to use them as parameters when creating personalities.

Whether and to what extent these properties are met by a specific implementation of GTA API in combination with a specific SE is out of scope of this document. The properties achieved by a specific implementation can be addressed by a security evaluation of the resulting system. Common evaluation processes are IEC 62443 (https://www.iec.ch/blog/understanding-iec-62443) and Common Criteria (https://www.commoncriteriaportal.org/).

At first, a foundation for defining SE properties is given by relevant protection goals that are potentially supported by SEs in some way. The following basic protection goals (PGs) are seen as relevant in the context of the GTA API and the underlying SEs:

- PG1a: Protect integrity of secrets, trusted attributes, and security metadata stored by SEs, against offline attacks or when powered off.
- PG1b: Protect confidentiality of secrets, which are stored by SEs, against offline attacks or when powered off.
- PG1c: Protect access to critical SE data by simple means, that is, based on identifiers, passwords, or platform configuration register (PCR) values.

AVA Documents
Document Sharing Hub

- PG1d: Protect access to critical SE data by sophisticated means, that is, based on corresponding GTA API access control mechanisms.
- PG2a: Protect against exposure of private keys, which are stored by SEs, during online or runtime attacks.
- PG2b: Protect against replay of critical SE communication messages, for example, by considering and enforcing cryptographic freshness.
- PG2c: Protect validation of trusted attributes, for example, compare password or verify public key operations, and corresponding access right decisions, which are done by an SE, against manipulation.
- PG2d: Protect operations affecting security metadata, especially transitions between device states.

NOTE Security metadata includes, for example, rules for transition between device states (cf. Figure 11)

Although implicitly grouped by their identifiers, protection goals can be treated individually. Nevertheless, a combination of protection goals is always possible.

How an SE achieves one of the protection goals can be confirmed by attestation that also uses the GTA API (cf. D.2).

From a technological point of view, the goals that are preceded by PG1 mainly correspond to an appropriately secure memory. The goals that are preceded by PG2 mainly correspond to a trusted execution environment. PG1c, PG1d, and PG2b need additional technological measures.

The following SE properties that are defined for the context of the GTA API are used to address the protection goals given above:

- SEPintegri: Protection of integrity of secrets, trusted attributes, and security metadata against offline attacks
- SEPintpers: Protection of integrity of a personality set, that is, the following mappings:
  - Between secret and trusted attributes
  - Between secret attributes, trusted attributes, and personalities
  - Between personalities and device states
- SEPintmeta: Protection of security metadata
- SEPseccrea: Securely create secret attributes on an SE
- SEPsecread: Ensure that secret attributes cannot be read from an SE when powered off
- SEPauthuse: Protection of use of secret and trusted attributes on an SE by using access tokens
- SEPauthman: Protection of management, for example, addition, of secret and trusted attributes by using device states
- SEPauthtru: Protection of validation based on trusted attributes
- SEPsecextra: Ensure that secrets cannot be extracted from an SE during operations that use them
- SEPsecrepl: Ensure that security-critical communication messages cannot be replayed

Table 2 gives a mapping between SE properties and protection goals.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Table 2 – Mapping between SE properties and protection goals

SEP	PG1a	PG1b	PG1c	PG1d	PG2a	PG2b	PG2c	PG2d
SEPintegri	X							
SEPintpers	X							
SEPintmeta								X
SEPseccrea		X						
SEPsecread		X						
SEPauthuse			X	X				
SEPauthman			X	X				
SEPauthtru							X	
SEPsecextra					X			
SEPsecrepl						X		

An SE property or a set of SE properties is characterized by Table 2. An SE can state to, for example, just achieve PG1c by SEPauthuse. An SE can announce to have further security properties regarding the protection goals mentioned or can even meet further protection goals. It can also be stated by an SE through attestation whether its security properties have been assessed or tested according to certain standards.

5.6 Implementation view

5.6.1 System design considerations

5.6.1.1 Secure element hardware-software co-design

An implementor of the GTA API can decide which functionality is provided by the actual SE hardware and which functionality is provided by the SE provider software.

Depending on the security requirements of the application, different types of SE and respective SE providers are possible:

- Software only, for example, to enable later migration to hardware based SEs.
- Hardware integrated, for example, firmware trusted platform module (TPM), trusted execution environment (TEE), system-on-chip (SoC) integrated, dedicated field programmable gate array (FPGA), security controller (SC), physical unclonable function (PUF), or hardware security module (HSM).
- Functional building blocks (for example, network element) that offer integrated security functionality (for example, authenticated network bulk encryption).

  This document does not mandate any specific technology for interfacing between the SE provider and the SE (cf. Figure 2). Vendors should make sure that the communication between SE provider and SE is sufficiently protected regarding the intended operational environment. Vendors should specify their assumptions for the intended operational environment. If specific technologies (for example, ISO/IEC 7816 Secure Messaging) are used which require keeping a secret at both ends of the communication channel, these technology details shall be handled completely within the SE provider. It is assumed that the IIoT application and GTA API (respectively the SE provider) are hosted within a common runtime address/process space. Following this assumption no extra security is achieved by coercing the application to care about the security of communication that is related to the SE.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

5.6.1.2 System security considerations

This document describes an interface. Its design was chosen to support secure use of an implementation of the GTA API. For the security of a system using such an implementation, a holistic approach is recommended. The security architecture of systems using a GTA API implementation should be created following a secure product development lifecycle, the result from a security risk assessment, and a list of relevant standard security requirements for a desired security level. For example, the International Standards [19], [20], [21], and [22] provide further guidance.

5.6.2 Personalities

5.6.2.1 Personality attributes

A personality is used within one or more specific security domains and enables the use of different security services to achieve, for example:

- Authentication of a GTA API instance as a whole or in part
- Verification of the authenticity of a GTA API external instance

Thus, a personality can include a specific secret that is used to prove a specific claimed identity. A personality can also include specific authentic information to verify if some secret is known by an external entity.

For the GTA API, a personality is defined using the following kinds of attributes:

a) Secret attributes

b) Trusted attributes

c) General attributes

The following examples are given to illustrate the personality concept.

The first example is a personality to be used within a public key infrastructure X.509 (PKIX) trust model which contains the following attributes:

- Secret attribute: An RSA private key used to compute signatures to prove the authenticity of the device.
- Trusted attributes: Peer public key certificates being unconditionally trusted by the device. These certificates include self-signed public key certificates, certification authority (CA) root certificates, and pinned peer public key certificates.
- General attributes: RSA public key certificate corresponding to the devices private key.

The second example shows a personality using an asymmetric crypto protocol without a public key infrastructure (PKI), for example, secure shell (SSH):

- Secret attributes: Set of private host keys, typically an RSA, a DSS, and an ECDSA key
- Trusted attributes: List of public keys for known hosts

The personality that is given in the last example is intended for a hybrid approach like a password authenticated Diffie-Hellman (DH) key exchange:

- Secret attribute: Password used by the device to authenticate itself.
- Trusted Attributes: Password verifier used to authenticate a password provided by a peer in a password-based authentication protocol.
- General attributes: DH group parameters to be used in a password authenticated key exchange.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Trust lists as defined by the OPC Foundation [15] are a combination of trusted attributes. Trust lists include information, such as trusted public key certificates or trusted certificate revocation lists, and general attributes.

Both secret and trusted attributes shall be protected according to the specified SE protection properties (cf. 5.5.8). An SE can provide an execution environment able to run operations using secret or trusted attributes as input. Confidentiality protection of trusted attributes is most probably not needed in contrast to confidentiality protection of secret attributes. Both types of attributes usually need integrity protection.

5.6.2.2 Personality fingerprint

Each personality shall define a well-defined process to compute a fingerprint on the personality. The fingerprint shall provide a cryptographically strong bond to the personality (for example, to refer to the personality in a cryptographic protocol to ensure cryptographic binding). In case the attributes relating to the personality change, the fingerprint shall also change.

Especially in case a personality is deleted and re-created using identical application provided identification attributes (for example, personality name), the fingerprint of the new personality shall be distinct from the deleted one. Otherwise, it would be possible to compromise the security by replacing a personality with a new personality which provides weaker security properties.

An example for a personality fingerprint is a cryptographic checksum (for example, based on SHA256) computed over a well-defined part of the value of the personality.

5.6.3 Profiles

5.6.3.1 Profile naming

Profiles shall be uniquely identified. GTA API uses string type profile names for this purpose. To avoid naming conflicts, the existing IANA Internet domain name registration system (DNS) shall be used for namespace management. Profile names shall use reverse domain name notation, that is, profile names start with a registered domain name in reverse order[2]. This notation allows any entity or organization holding a registered domain name to specify a profile, for example:

- ch.iec.30168 for profiles defined within this document
- org.opcfoundation for profiles defined within the scope of OPC UA
- org.ietf.tls13 for profiles defined within the scope of transport layer security (TLS) 1.3

The naming scheme can be used to define subprofiles, for example:

- org.ietf.tls13.x509 for X.509 public key certificate-based TLS
- org.ietf.tls13.psk for PSK (pre-shared keys) based TLS

5.6.3.2 Personality creation and deployment profiles

Personality creation and deployment profiles are used during the creation and deployment of new personalities.

The properties that shall be determined by a personality creation and deployment profile are defined in Table 3 and Table 4. Profile definitions shall provide any mandatory (m) properties. Optional (o) attributes can be specified to provide additional information.

---

2 See https://en.wikipedia.org/wiki/Reverse_domain_name_notation

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Table 3 – Properties of personality creation profiles

Property	Mandatory (m)/ Optional (o)	Description
Security Mechanism	m	Type and strength of the security mechanism that is supported by the personality (for example, AES256 secret key, EC25519 private key)
Fingerprinting	m	Scheme or algorithm used to derive a unique personality fingerprint (5.6.2.2)
Attributes	o	Well-defined attributes of the personality
Usage Info	o	Reference to enrollment (5.6.3.3) or usage profiles (5.6.3.4) that is supported by the resulting personality

Table 4 – Properties of personality deployment profiles

Property	Mandatory (m)/ Optional (o)	Description
Security Mechanism	m	Type and strength of the security mechanism that is supported by the personality (for example, AES256 secret key, EC25519 private key)
Import Format	m	Type and format of the archive data that are used to represent the personality being imported (for example, PKCS#12)
Fingerprinting	m	Scheme or algorithm that is used to derive a unique personality fingerprint (5.6.2.2)
Attributes	o	Well-defined attributes of the personality
Usage Info	o	Reference to usage profiles (5.6.3.4) that are supported by the resulting personality

5.6.3.3 Personality enrollment profiles

Personality enrollment profiles are used during the enrollment and registration of personalities using a trusted third party (for example, an X.509 certification authority).

The properties that shall be determined by a personality enrollment profile are defined in Table 5. Profile definitions shall provide any mandatory properties. Optional attributes can be specified to provide additional information.

Table 5 – Properties of personality enrollment profiles

Property	Mandatory (m) Optional (o)	Description
Profile Dependencies	m	The profile may require that the personality has specific properties that are typically specified during personality creation. These dependencies can be expressed by referencing to a specific personality creation profile. It is also possible to determine an appropriate creation profile from the specification of the supported security mechanisms.
Enrollment Attributes	m	Attributes (including attribute format) that shall be provided to build the enrollment request (for example, an X.509 subject name, requested X.509 public key certificate extensions)
Enrollment Artifact	m	Format of the enrollment artifact (for example, PKCS#10)

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

GTA API profiles enable service providers for X.509 certification/registration authorities to offer services that are aligned with enrollment profiles defined in association with GTA API. In this case the profile can also consider properties which are not directly related to GTA API. One example property is the template used for the resulting X.509 certificate such as subject name attributes, validity, or certificate extensions. This offering would allow users to decide whether a specific CA service is suitable for an intended application based on the profile name as single criterion.

5.6.3.4 Personality usage profiles

Personality usage profiles are used while applying personalities for security operations. Examples for security operations are creating a signature, performing encryption or decryption, computation of cryptographic checksums, verification of secret passwords.

The properties that shall be determined by a personality usage profile are defined in Table 6. Profile definitions shall provide any mandatory properties. Optional attributes can be specified to provide additional information.

Table 6 – Properties of personality usage profiles

Property	Mandatory (m) Optional (o)	Description
Profile Dependencies	m	The profile can require that the personality has specific properties, typically specified during personality creation or deployment. These dependencies can be expressed by referencing to a specific personality creation or deployment profile. It is also possible to determine an appropriate creation or deployment profile from the specification of the required security mechanisms.
Supported Functions	m	Functions of GTA API supported for this profile. It is possible that some profiles will not be applicable for use with all functions of GTA API. For example, it is possible that a profile that is destined to support the establishment of secure channels between two parties will not work with data protection functions.
Usage Attributes	m	Extra attributes required for usage of the personality, for example, a challenge value required within a challenge-response protocol
Usage Artifact	m	Type and format of the artifact resulting from an operation involving the personality (for example, PKCS#1); Depending on the profile, multiple artifacts can result while using a personality according to the given profile.

5.6.4 Device states

5.6.4.1 General

GTA API distinguishes two types of device states:

- Owner states: The device is in the ownership of one actor along the value creation chain (for example, the machine builder).
- Transition states: The device is in transition from one owner to the succeeding owner. The transition states are created by the originating owner to allow transition to the next owner in the chain. GTA API allows testifying the genuineness of a device transition state. Successful attestation of a transition state allows the new owner to decide whether the device is still in the state that was intended by the originating owner. A failed attestation indicates that someone else already tampered with the device. Once the transition state gets verified, the recipient takes ownership of the device by installing the owner state of the recipient on top of the latest transition state.

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

The organization of device states can be looked at as a stack. Two operations are possible on the device state stack (cf. Figure 10):

- Advance: Pushes a new state on top of the existing stack. The new state on top of the stack is either a transition state or an owner state.
- Recede: Pop states from the top of the stack until the next transition state is reached. All personalities belonging to ownership states that are affected by the operation are discarded.

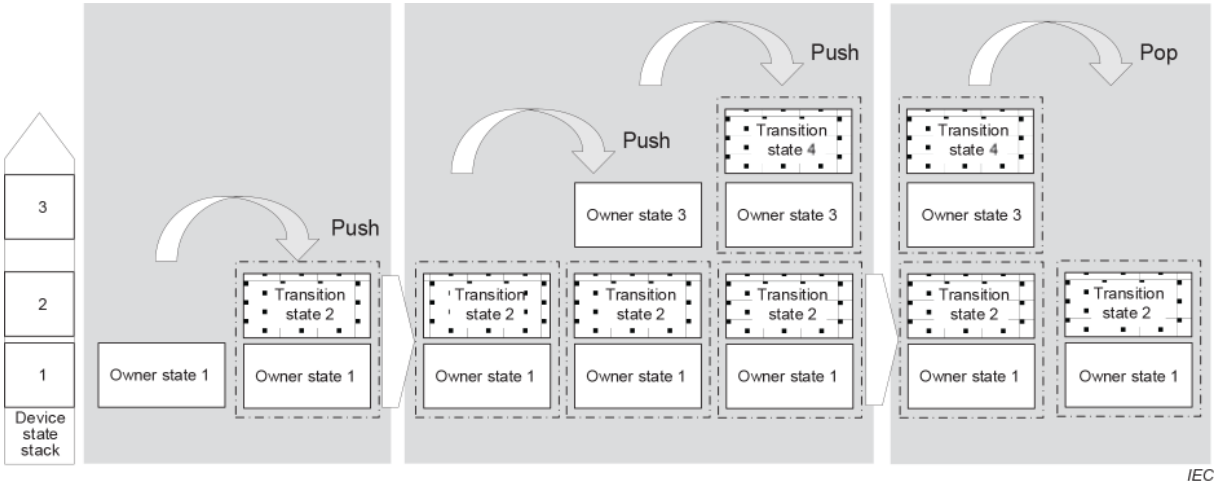

Figure 10 – Device state stack

Push and pop operations result in transitions between device states as shown in Figure 11.

A new device state is pushed automatically whenever a new personality is created from either the initial state (cf. 5.6.4.2) or a transition state.

Transition states are created by an explicit invocation of a `gta_devicestate_transition()` operation. This operation also allows definition of an access policy which shall be met to remove the transition state from the stack:

1) `PHYS`: The removal requires physical presence (cf. 5.6.5.2.5).
2) `PHYS|CREATOR`: The removal requires either physical presence or explicit authentication as defined by the state creator.
3) `CREATOR`: The removal requires explicit authentication as defined by the state creator, that is, the state cannot be removed using physical presence.

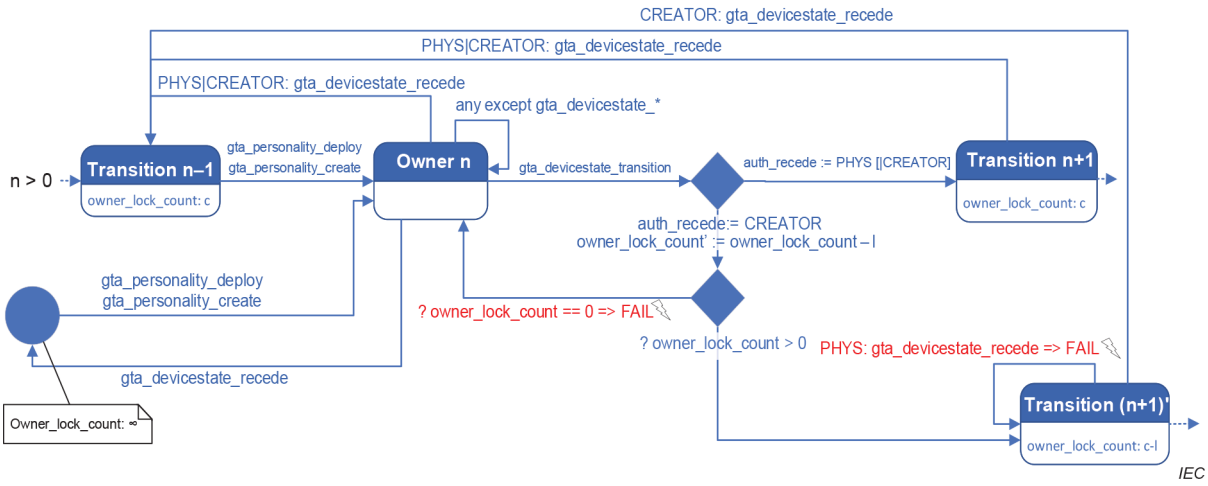

Figure 11 – Device state transitions

Documents
Document Sharing Hub

The use of option 3) can be restricted by an additional parameter `owner_lock_count`. `owner_lock_count` determines whether creator-defined-only authentication can be specified for future states. Whenever a new transition state is pushed specifying option 3), the value that is specified for `owner_lock_count` shall be smaller than for preceding push operations. In case no smaller value exists (that is, `owner_lock_count == 0`) option 3) can no longer be used, that is, all subsequent states can be removed using physical presence.

The following example illustrates how GTA API supports the transition of ownership between different entities.

Figure 12 shows the transition from the device manufacturer to the machine builder. Initially the device is in state 1. State 1 is used to provision the personalities of the manufacturer to the device. After that the device manufacturer pushes a transition state. The machine builder verifies the genuineness of the transition state and continues installing device state 3. Device state 3 is owned by the machine builder and used to install the manufacturer personalities.

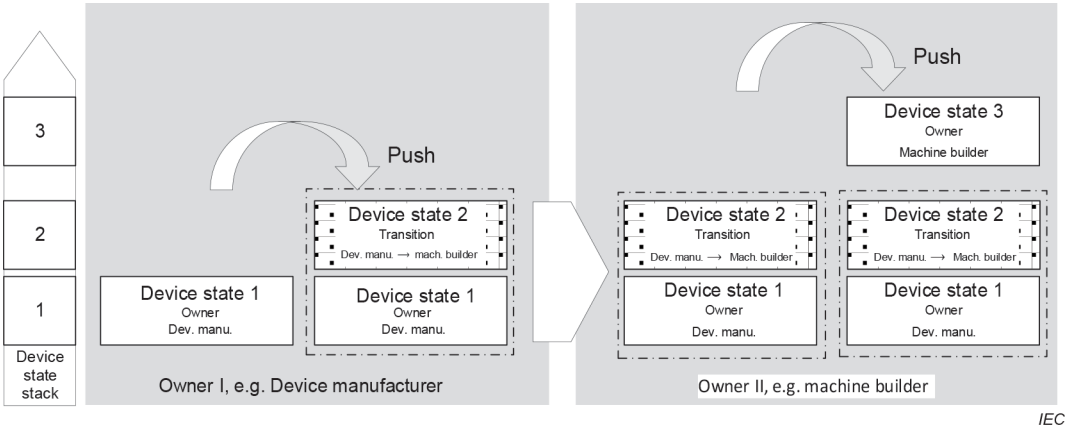

Figure 12 – Device state stack (push)

Figure 13 illustrates how the device is returned to the device state owned by the machine builder after a machine has been returned from operation (decommissioning). Typically, the machine builder does not know any secret attributes that have been installed by the operator. However, it is possible that the machine builder is still allowed to use physical presence to pop the device state that has been installed by the operator. This operation returns the device to the device state created by the machine builder.

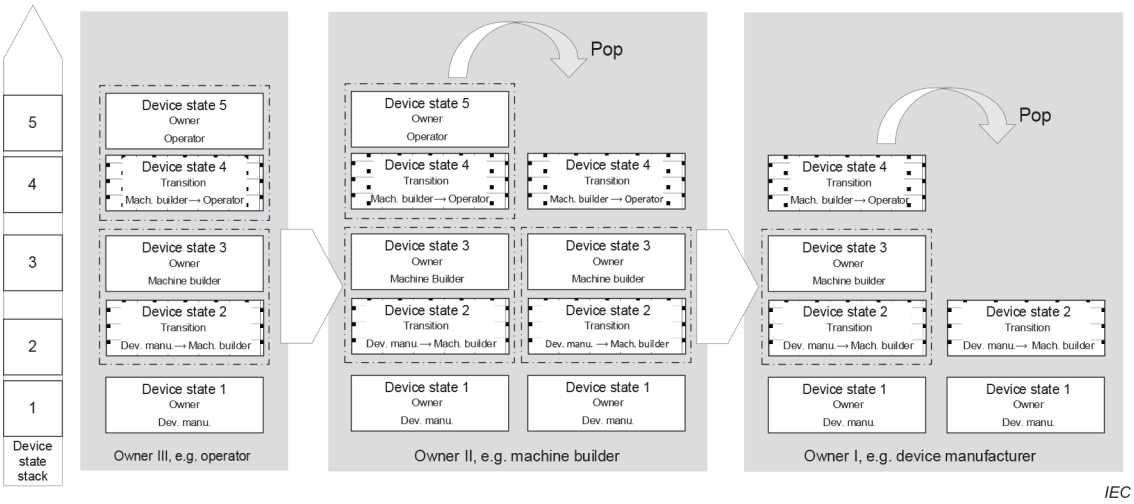

Figure 13 – Device state stack (pop)

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

After the last pop operation permitted by GTA API, one residual state which cannot be popped remains on the stack. Whether pushing the first device state requires some means of authentication is left to the implementation (cf. 5.6.4.2).

5.6.4.2 (Pre-)Initial device state

The initial device state of the GTA API from a security management point of view strongly depends on its specific integration and initial configuration of SEs. It includes:

- Primary access conditions
- Primary personalities (for example, including trusted attributes for further bootstrapping)

Primary access conditions and in primary personalities may be empty after manufacturing or even at later stages. Apart from the device state, that is, access conditions and provided personalities, the functionality of the GTA API is also determined by the number of different profiles offered. Security levels are understood as part of the initial device state of the GTA API. But because security levels are static attributes of the functionality that is offered by the GTA API, they are not explicitly covered in 5.6.4.2.

The overall trustworthiness of the initial state of the GTA API depends on the following presuppositions:

- Pre-initial state of the component or device
- Pre-initial states of all SEs at the time of the first installation of the GTA API
- Physical and logical protection of the component or device that serves as installation target for the GTA API
- Physical and logical protection of the physical environment in which the component or device is used at the time of the first installation of GTA API
- Protection of the GTA API installation files before they are used and at their time of usage
- Protection of any further item that is later used for contributing to the initial state of the GTA API, for example, states resulting from secure boot

The initial device state of the GTA API can be created based on different trust models. Here, the following two fundamentally different trust models are considered:

a) Trust on first use (TOFU), also referred to as "resurrecting duckling"

NOTE "Resurrecting duckling" is a security policy mainly based upon the two states "imprintable" and "imprinted". An imprintable device accepts any security configuration from any new owner. An imprinted device acts according to the security configuration from its owner. F. Stajano and R. Anderson [29] use the definition: "the resurrecting duckling security policy model [...] describes secure transient association of a device with multiple serialised owners". The latter essentially means that a device can recede to the state imprintable.

b) Trust that is based on already possible protection based on personalities

Trust that is based on already possible protection that is based on personalities is only possible if at some earlier point in time TOFU was performed. A common place for TOFU for a GTA API instance is during manufacturing of the component that uses the GTA API. As preparatory step, manufacturers of SEs can perform TOFU on their own. This step provides the component or device manufacturer with a way to authenticate against the SEs on first usage. But TOFU can also be performed by other players like distributors, machine builders or operators. After TOFU has been performed once, all following steps can be designed in a way that allows to build up a full chain of trust. This design relies on a consistent use of personalities and access policies that have been provided by the first TOFU step. A high level of trustworthiness can be reached when TOFU is performed during manufacture of SE and GTA API component or device and when all subsequent steps are consistently implemented in secure and reliable ways on top of this first step.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

As an example, trust that is based on manufacturer protection and personalities including manufacturing of SE is now described further and briefly elaborated on. Other trust models can be used as well as other ways to extend the trust provided by manufacturer personalities.

Assume some component that offers applications that in turn use the GTA API together with exactly one SE in form of a security controller. Further assume that the security controller is manufactured by one manufacturer and the rest of the component is manufactured by another manufacturer. The manufacturer of the security controller is referred to as ‘manufacturer SC’, that is, manufacturer security controller. The manufacturer of the rest of the component is referred to as ‘manufacturer CO’, that is manufacturer component. Manufacturing includes physical and logical parts, that is, hardware and software or firmware. In this setting, manufacturer SC uses a protected manufacturing environment to store some secret encoded as reference value within the secure memory of the security controller. The security controller gets locked by manufacturer SC and can only be opened using this secret. Using some previously established trusted communication channel or trusted delivery path, manufacturer SC delivers the secret or the corresponding security controller to manufacturer CO. Manufacturer CO integrates this security controller within a protected manufacturing environment. Here, integration means hardware assembly of the component. In this assembly process the security controller is one element that gets integrated together with first installation of the GTA API from a protected software repository. During GTA API installation, the secret for unlocking the security controller is provided as well. The GTA API is used by a corresponding application to first unlock the security controller. The application then generates an asymmetric cryptographic key pair on the SE together with a certificate signing request for the public key part. As last step the application imports a manufacturer X.509v3 public key certificate together with corresponding trusted attributes. All artifacts created by the application altogether serve as initial device identifier (for example, initial device identifier (IDevID) according to IEEE 802.1AR [10]). During this process, the GTA API pushes Device State 1 on its stack representing the IDevID with manufacturer CO as owner. The IDevID is the first personality that can be used by the GTA API, for example, to update the GTA API itself under control of an update application. Such updates can include updates for SE and SE provider whereby security can in turn be based on the pre-initial states of the SE. Next, component management pushes Device State 2 as transition state on the GTA API device state stack. Again, a secret attribute can be created and stored within the security controller in a way that unlocking of the GTA API is only allowed after proving knowledge of this secret by an application, which can again include setting of trusted attributes. As an alternative, an application is provided in a way that Device State 2 is only allowed to be used once by the next owner. That means, multiple use of the transition state is not allowed and can be detected. Setting of Device State 2 concludes the GTA API specific runtime part of manufacturing. All further transitions can as well be protected using adequately created transition states or device states. Delivery of unlocking secrets, if used, from one owner to the next should be protected using appropriately trusted out-of-band ways. This way, new device states can only be pushed onto the device state stack by respective component users. If a transition state can only be used once, a new owner can detect whether the GTA API is still in this state. If transition states are created without any protection, the GTA API can be configured further by anyone who has adequate access to the component that hosts the GTA API.

As a complement, initial access that always starts empty either stays empty or is based on a specific set of previous operations. Conducting secure boot is an example for such operations. The latter means that depending on reference values for unlocking secrets, software or firmware image hashes, initial access can be achieved during start-up of the component or device.

After all steps mentioned before were performed, the component or device with integrated GTA API has an initial and potential one or more further device states of the GTA API. These device states include personalities and sets of secret and trusted attributes. There are defined conditions for initial access of GTA API access control depending on functionalities of the component during start-up. Summed up, beginning with manufacturers SC and CO all subsequent owners can use the GTA API in a trustworthy way that is determined by the security policy of the preceding owner.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Refer to 5.6.4.1 and 5.6.5 for further descriptions on how device states and initial access, respectively, are reached or managed. Also refer to Clause E.1 for a specific example.

5.6.5 Access control

5.6.5.1 Initial access

Initial access shall be available automatically after successful start-up of the system (including GTA API and SE). The conditions which apply to grant initial access are implementation dependent. These conditions shall be documented for the GTA API user developing an application on the device. Examples for such conditions are:

- Completion of a secure boot sequence
- Completion of specific system runtime integrity check or system environment integrity check, or both.

Initial access can be void if there are error conditions that are detected by either GTA API or the SE. The respective error conditions are implementation-specific and shall be documented.

5.6.5.2 Access tokens

5.6.5.2.1 General

All access policies that extend beyond initial access are granted using access tokens. Figure 14 illustrates how an access token is built to incorporate several properties.

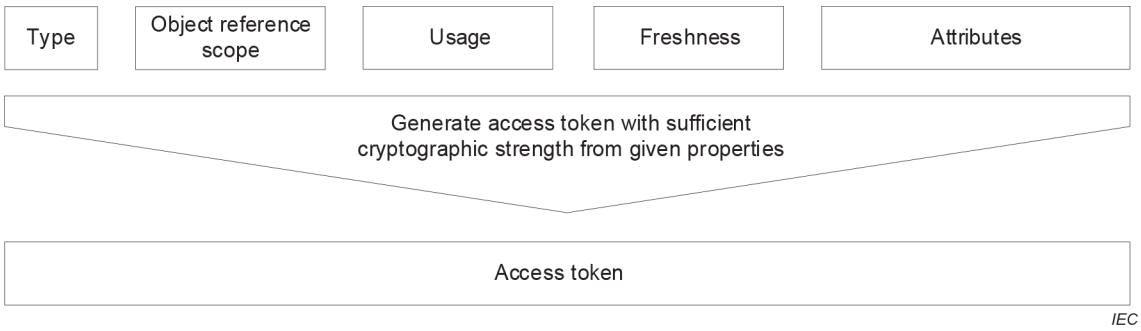

Figure 14 – Access token

An access token confirms the following properties.

- Type: There can be different types of access tokens. Access tokens allow to convey privileges among local processes or allow to grant privileges that are held by users or remote services. Access token types are described in detail in 5.6.5.2.
- Object Reference/Scope: A reference to the object or a set of objects for which the token is valid (cf. Table 1).
- Usage: The kind of permission granted (cf. Table 1).
- Freshness: Information allowing to determine whether the token is still valid. The validity of an access token shall not persist after restart of the device.
- Attributes: Extra attributes which allow the SE to determine whether the token is valid for the intended operation. Each different type of access token contains specific attributes. Attributes can be used to tie the applicability of a token to precisely defined operations. An example for such an operation is the update of a trusted attribute to a specific value. Tying tokens to operations is especially useful for personality derived access tokens (cf. 5.6.5.2.4).

The access token grants the right to use the specified object for the specified use to the bearer of the token.

AVA Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

The access token itself is a 256 bit value. It shall not be possible to predict the value of an access token. It shall also not be possible to guess the value of an access token with more than negligible probability. The access token shall provide a cryptographic strength of at least 128 bit. The internal structure of the access token is left to the specific implementation. Exchange of access tokens between SEs of different vendors is not required.

GTA API uses different types of access tokens to allow flexible definition of access policies (5.6.5.3).

5.6.5.2.2 Basic access tokens

To enable implementation of advanced access control between different applications, GTA API implementations can support basic access tokens. Basic access tokens enable implementing access control between different applications that can share a single SE (cf. 5.5.1.3). Basic access tokens are issued by the SE containing the respective objects.

Basic access tokens have no additional attributes. Basic access tokens can be retrieved from GTA API after system start-up until the function to retrieve access tokens is actively disabled. After the function is disabled, it will not become available again until the system (including the SE) is restarted. This behaviour is realized using a dedicated token issuing token (see 5.6.5.2.3).

5.6.5.2.3 Token issuing token

GTA API implementations supporting basic access tokens shall support granting of basic access tokens using a token issuing token. Basic access tokens can only be issued after authentication with a dedicated token issuing token. The token issuing token shall be provided only once during start-up of the device (cf. 5.6.6).

This procedure can be used to separate access between different applications. A privileged process run at system start-up gets hold of the token issuing token and requests basic access tokens for various objects that are protected by the SE. Either all required tokens are requested upfront and passed on to dependent processes or the privileged process provides a service to acquire access tokens. This privileged service should be protected by appropriate access control that is based on OS mechanism. The further distribution of the basic access tokens to subsequent applications is subject to the power of decision of the privileged process. Process privilege management and process isolation is the responsibility of the system OS.

5.6.5.2.4 Personality derived access tokens

GTA API implementations can support personality derived access tokens. Personality derived access tokens result from the successful use of a personality. Depending on the profile used, there are various possibilities for personality derived access tokens, for example:

- Dedicated personalities and profiles that are used to enable simple password or passcode-based authentication schemes (cf. ch.iec.30168.basic.passcode, Clause B.1).
- Dedicated personalities and profiles that are used for more complex authentication schemes, for example, challenge response based on symmetric or asymmetric secrets.
- Access tokens resulting from a side-effect of a general cryptographic protocol. For example, the successful verification of a digital signature or successful completion of a cryptographic key exchange can also yield an access token. This access token can then be used to authenticate further operations.

The attributes of a personality derived access token should include at least binding to the personality and the profile leading to the creation of the access token as illustrated in Figure 15. Other attributes can be used to further restrict the applicability of the access token. For example, the verification of a signature using a specific personality can result in an access token. The resulting access token then grants the privilege to set a trusted attribute of that personality to that exact value authenticated by the signature.

AVA Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

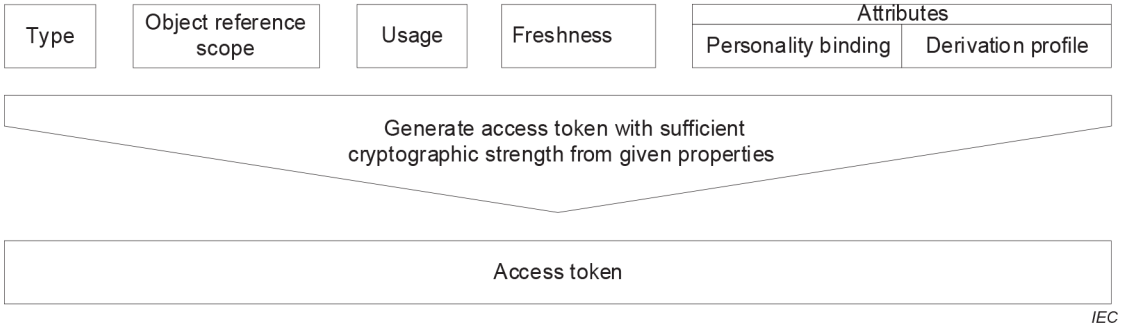

Figure 15 – Personality derived access token

It is possible to set up a chain of authentication protocols depending on different personalities to require multiple authentications before using a personality.

EXAMPLE A personality according to the profile ch.iec.30168.basic.passcode (Clause B.1) is used to create an access token resulting from the verification of a password. This access token is used to protect the usage of another personality according to the same profile ch.iec.30168.basic.passcode. The verification of the second password is only possible after providing the first password. The authentication token that is obtained upon verification of the second password is used to allow computation of an authentication artifact (for example, a digital signature). Thus, computation of the authentication artifact will only be possible after both passcodes are made available.

5.6.5.2.5 Physical presence access tokens

Physical presence access tokens can be acquired during device start-up. Physical presence tokens shall only be issued by GTA API if the device (and SE) has evidence that physical presence criteria are fulfilled (for example, keylock), cf. 5.6.6. Definition of these criteria is solution specific and out of scope of this document. Vendors shall describe these criteria for their respective solution.

The security impact that is achieved by privileges granted by physical presence shall not exceed the security impact achievable by an entity (potential attacker) exercising complete physical control over the device (for example, with a sledgehammer). Especially availability can be affected by privileges acquired by physical presence. For example, availability related privileges include deletion of secret or trusted attributes but not their creation or modification.

5.6.5.3 Access policies

Access policies are used to describe the rules which apply to allow the execution of a specific operation (cf. Table 1) on an object.

Access policies shall comply with the rules shown in Figure 16 in Backus–Naur form[3] (BNF).

[3] See, for example, https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

```
<Access Policy> ::=
          <Personality Access Policy>
        | <Device State Access Policy>
<Personality Access Policy> ::=
          <Initial Access>
        | <Advanced Access Policy>
<Advanced Access Policy> ::=
         <Basic Access Token Descriptor>
        | <Basic Access Token Descriptor> <Personality Derived Access>
        | <Personality Derived Access Token Descriptor> <Personality Derived Access>
<Personality Derived Access> ::=
       <Personality Derived Access Token Descriptor> <Personality Derived Access>
     | <Empty>
<Device State Access Policy> ::=
          <Physical Presence Access Token Descriptor> <Personality Derived Access>
        | <Personality Derived Access>
```

IEC

Figure 16 – Access policy composition (BNF)

That means an access policy for a personality requires initial access (cf. 5.6.5.1) or the presentation of a basic access token (cf. 5.6.5.2.2) or personality derived access token (5.6.5.2.4). There can be an optional choice of further personality derived access tokens. Access policies shall be evaluated as disjunctions ("or"). Access to a personality that is protected by a respective access policy is granted in case the application presents at least one access token according to the policy.

It is also possible to require multiple access tokens before an operation is granted ("and", conjunction). Conjunctions can be built by chaining personality derived access policies (cf. 5.6.5.2.4). For example, to require two distinct passcodes before usage of a personality holding an RSA private key, usage of the RSA key personality is protected by a first passcode personality. Verification of this first passcode is protected by a second passcode personality.

To fulfil the access policy for a device state, a physical presence access token or, optionally, a personality derived access token can be presented.

5.6.6 GTA API start-up

Granting of several access permissions depends on the start-up phase of the GTA API device (cf. Figure 17). Transition between GTA API start-up states is controlled by the runtime environment typically in relation to the boot phase of the device.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

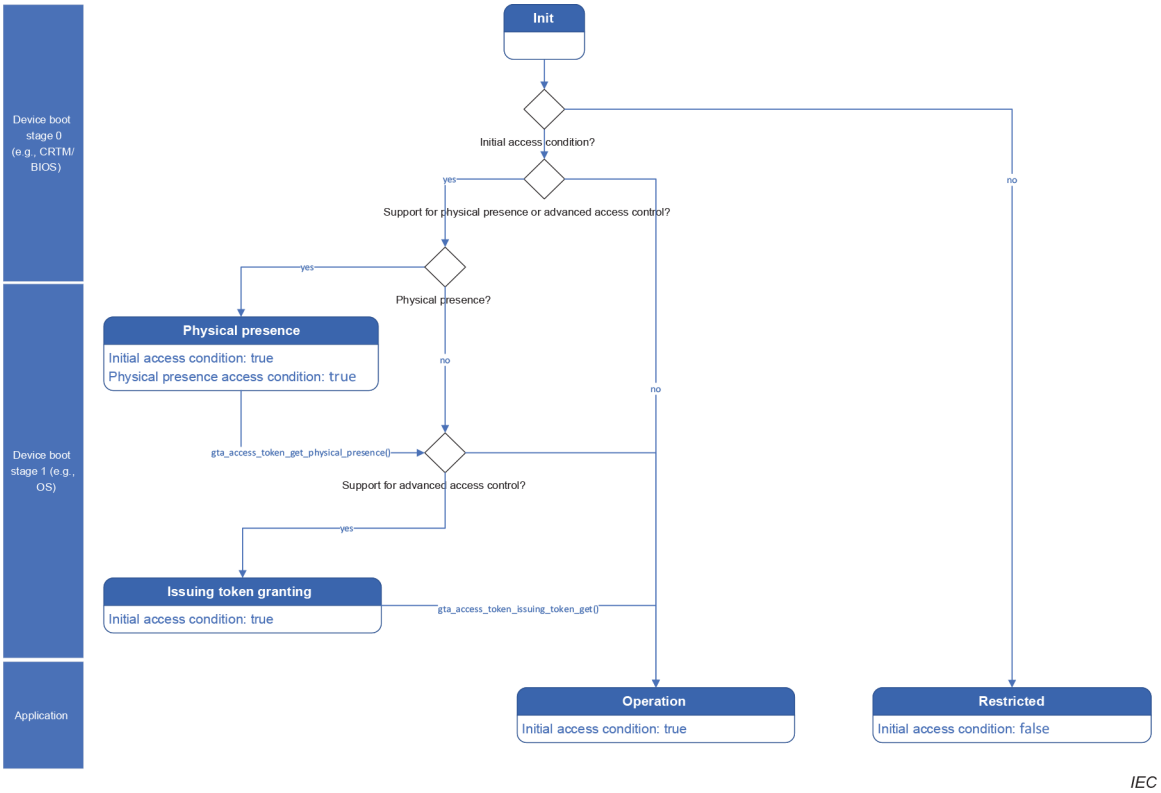

Figure 17 – GTA API start-up phases

GTA API recognizes the following start-up states:

- Init: This is the first state that is reached after system start-up. GTA API only provides limited functionality in this state. The runtime system decides whether the prerequisites to grant initial access and physical presence are met or not. The start-up phase of GTA API is advanced accordingly.
- Physical presence or no physical presence: Depending on a decision of the runtime environment, GTA API and the underlying SE either permit or reject operations requiring physical presence authentication (cf. 5.6.5.2.5). These start-up phases may be omitted in case the device does not support physical presence.
- Issuing token granting: This state is intended for a privileged process of the runtime system. The privileged process can get hold of the token issuing token (cf. 5.6.5.2.3) and collect basic access tokens (cf. 5.6.5.2.2). The basic access tokens are distributed by the privileged process among subsequent application processes. Application processes shall not be started before the token issuing token has been collected by the privileged process.
- Operation: This state covers the general usage of GTA API by applications.
- Restricted: This state covers the general usage of GTA API by applications with limited functionality if the initial access condition is not met. Especially it is not possible to use any functions which are subject to access control in this state.

The start-up phase state is system wide, that is, it is shared among all instances of GTA API running on a device. Resetting the GTA API start-up phase requires a power cycle of the device.

6 API specification

6.1 Overview

GTA API is structured into several function groups. Each function group addresses a specific aspect being relevant for different user types of GTA API. See Table 7.

Documents
Document Sharing Hub

Table 7 – GTA API function groups

Function group	Subclause	Description
General management functions	6.6.2	General functions to manage GTA API
Process synchronization	6.6.3	Functions to be used by GTA API for process synchronization. This interface shall be provided by the application. The platform vendor can provide appropriate functions to the application developer
Secure memory management	6.6.4	GTA API utility functions for secure memory management which can also be used by the application
Function parameter I/O streams	6.6.5	Parameters to and results from GTA API are passed using streams. The application developer can implement a stream interface according to the needs of the application. Platform vendors or providers of GTA API can provide reference implementations
Instance management functions	6.6.6	Functions to initialize and finalize GTA API instances to be used by the application
Context management functions	6.6.7	Functions to establish GTA API contexts that are used to organize the execution of GTA API functions
Access token functions	6.6.8	Functions to acquire access tokens required to authenticate for the use of privileged functions on protected GTA API objects
Device state management functions	6.6.9	Functions to manage the state of the device/SE
Identifier and personality management	6.6.10	Functions for the management (creation, deletion) of identifiers and personalities
Access policy management functions	6.6.11	Functions to create access policies to be attached to GTA API objects (for example, personalities)
Data protection functions	6.6.12	Functions to protect data
Channel protection functions	6.6.13	Functions to support the establishment of secure channels
Supplementary security functions	6.6.14	Security functions which do not relate to specific GTA API objects, for example, secure random numbers
Trusted execution environment	6.6.15	Functions to interface with the trusted execution environment which can optionally be provided by an SE
Secure element provider implementation support	6.6.16	Functions to support the implementor of SE providers to interface with the GTA API framework

Different implementations of GTA API can provide different extent of support for the individual functions and related features that are described in this document.

Table 8 provides an overview on the feature classes being distinguished. The description explains how an implementation of GTA API shall react in case a feature or functionality is not supported.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Table 8 – GTA API feature classes

Feature class	Description
T	Threading: Shall be supported in case GTA API is supposed to be used in parallel/multithreaded environments. Implementations not supporting parallel environments shall implement these functions as stubs. Calling these stub functions shall not result into an error condition or other side-effects.
U	Update: Dynamic update of the GTA API itself. In case the feature is not supported by the GTA API implementation, the functions shall fail with GTA_ERROR_FEATURE_NOT_SUPPORTED.
A	Advanced access control: Support of advanced access control. In case the feature is not supported by the GTA API implementation, the functions shall fail with GTA_ERROR_FEATURE_NOT_SUPPORTED.
X	Trusted eXecution: Support of trusted execution. In case the feature is not supported by the GTA API implementation, the functions shall fail with GTA_ERROR_FEATURE_NOT_SUPPORTED.
C	Create: This function can be omitted in case only the usage of static, preinitialized, and prepersonalized personalities is supported by the GTA API implementation. GTA API implementations which do not support this functionality shall fail with GTA_ERROR_FEATURE_NOT_SUPPORTED.
M	Mandatory: This function belongs to the mandatory subset of GTA API and shall be supported by all implementations.
P	Profile-specific: Supported depending on the used profile. The function shall behave according to the specification provided by the selected profile. In case the behaviour of the function is not defined by the respective profile, it shall fail with GTA_ERROR_PROFILE_UNSUPPORTED.

‘C’ (Create) takes precedence over ‘P’ (Profile-specific). That means GTA API implementations that do not support creation of personalities and attributes cannot support any profiles specifying these operations and shall fail with GTA_ERROR_FEATURE_NOT_SUPPORTED.

Table 9 specifies a feature class for each function of GTA API. Some of the features are optional. Independent of the feature class of a function, GTA API framework shall be able to address and exploit the full functionality of subordinate modules.

The function category indicates whether the behaviour of the function is inherent to the GTA API framework core, the host platform GTA API is running on, or the individual SE providers.

Table 9 – GTA API functions per feature class

Function	Feature class	Category
General management functions		
gta_library_info	M	Framework core
gta_register_provider	M	Framework core
gta_update_library	U	Framework core
Process synchronization		
gta_mutex_create	T	Host platform
gta_mutex_destroy	T	Host platform
gta_mutex_lock	T	Host platform
gta_mutex_unlock	T	Host platform
Secure memory management		
gta_secmem_calloc	M	Host platform
gta_secmen_checkptr	M	Host platform
gta_secmem_free	M	Host platform

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 30168:2024

Function	Feature class	Category
Instance management functions		
gta_instance_final	M	Framework core
gta_instance_init	M	Framework core
Context management functions		
gta_context_auth_get_challenge	P	SE provider
gta_context_auth_set_access_token	A	SE provider
gta_context_auth_set_random	P	SE provider
gta_context_close	M	Framework core
gta_context_open	M	Framework core
gta_context_get_attribute	P	SE provider
gta_context_set_attribute	P	SE provider
Access token functions		
gta_access_token_get_basic	A	SE provider
gta_access_token_get_pers_derived	P	SE provider
gta_access_token_get_issuing	A	SE provider
gta_access_token_get_physical_presence	A	SE provider
gta_access_token_revoke	A	SE provider
Device state management functions		
gta_devicestate_attestate	P	SE provider
gta_devicestate_recede	C	SE provider
gta_devicestate_transition	C	SE provider
Identifier and personality management		
gta_identifier_assign	C	SE provider
gta_identifier_enumerate	M	SE provider
gta_personality_add_attribute	C, P	SE provider
gta_personality_add_trusted_attribute	C, P	SE provider
gta_personality_attributes_enumerate	M	SE provider
gta_personality_attestate	P	SE provider
gta_personality_create	C	SE provider
gta_personality_deploy	C	SE provider
gta_personality_enroll	C, P	SE provider
gta_personality_enroll_auth	C, P	SE provider
gta_personality_enumerate	M	SE provider
gta_personality_enumerate_application	M	SE provider
gta_personality_get_attribute	P	SE provider
gta_personality_remove_attribute	P	SE provider
gta_personality_remove	C	SE provider
gta_personality_deactivate	P	SE provider
gta_personality_activate	P	SE provider
gta_personality_deactivate_attribute	P	SE provider
gta_personality_activate_attribute	P	SE provider

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

Function	Feature class	Category
Access policy management functions		
gta_access_policy_add_basic_access_token_descriptor	A	Framework core
gta_access_policy_add_pers_derived_access_token_descriptor	A	Framework core SE provider[a]
gta_access_policy_add_physical_presence_access_token_descriptor	A	Framework core SE provider[a]
gta_access_policy_create	A	Framework core
gta_access_policy_destroy	A	Framework core
gta_access_policy_get_access_descriptor_attribute	M	Framework core
gta_access_policy_get_access_descriptor_type	M	Framework core
gta_access_policy_simple	M	Framework core
gta_access_policy_enumerate	M	Framework core
Data protection functions		
gta_authenticate_data_detached	P	SE provider
gta_seal_data	P	SE provider
gta_unseal_data	P	SE provider
gta_verify_data_detached	P	SE provider
gta_verify	P	SE provider
Channel protection functions		
gta_seal _message	P	SE provider
gta_security_association_accept	P	SE provider
gta_security_association_destroy	P	SE provider
gta_security_association_initialize	P	SE provider
gta_unseal_message	P	SE provider
Supplementary security functions		
gta_get_random_bytes	M	SE provider
gta_attestate	P	SE provider
Trusted execution environment		
gta_trustex_function_install	X	SE provider
gta_trustex_function_uninstall	X	SE provider
gta_trustex_function_execute	X	SE provider
gta_trustex_function_terminate	X	SE provider
Secure element provider implementation support		
gta_provider_context_close	M	SE provider
gta_provider_context_open	M	SE provider
gta_provider_get_params	M	Framework core
gta_context_get_provider_params	M	Framework core
gta_context_get_params	M	Framework core

[a] If this function is used with GTA API, the used SE providers must support the created access policy. SE providers shall not permit access to objects protected by access policies not supported by the SE provider.

Annex F provides some additional guidance for the implementors of GTA API.

Annex G provides some examples for the usage of GTA API.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

6.2 Language binding

Throughout this document C99 syntax is used to specify the API.

GTA API uses the following definitions of the C standard library:

- `stdbool.h`
- `stdint.h`
- `stddef.h`

It is possible that other language bindings will be specified in the future.

6.3 Endianness

Implementations shall use the endianness of the underlying host system to represent basic C types (int, char, …). Whenever bytestrings are used to represent big numbers or other cryptographic artifacts, these bytestrings shall be in big endian. A profile (cf. 5.5.5) can explicitly require little endian encoding.

6.4 Exception handling

GTA API implementations shall be designed to never fail in a way that results in cryptographic compromise, for example:

- Wrong handling of total failure of random number generator
- Exposition of plaintext due to unwanted equality of input and output buffer
- Enablement of side-channel analysis through observation of exception handling

6.5 Using GTA API from an application

6.5.1 Header files

A typical C application imports API functionality by C header files. Each implementation of GTA API shall ensure that it can be interfaced with by an application using the header files given in Annex A. Implementations of GTA API may provide their own header files if these do not break compatibility with this document.

6.5.2 Call conventions and error handling

All functions that are defined within GTA API follow the following schema:

```
[bool|void *] gta_function([...,] gta_errinfo_t * p_errinfo);
```

The boolean interpretation of the return code tells whether the function completed successfully or not. If a function fails, the value that is referenced by p_errinfo can be used to obtain further information about the type of error. If a function completes successfully, the value that is referenced by p_errinfo is left unmodified by the GTA API function.

The following convention should be followed when calling a GTA API function:

```
gta_errinfo_t errinfo;
[void * p_res;]

if ([p_res = ] gta_function([...,] &errinfo)) {
  do_something();
}
else {
  do_error_handling(errinfo);
}
```

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

6.6 Types and function documentation

6.6.1 Basic types

6.6.1.1 Description

These are the basic types which are used throughout GTA API.

6.6.1.2 Defines documentation

6.6.1.2.1 GTA_ERROR_ACCESS

#define GTA_ERROR_ACCESS 15

Access to the function or object is not granted.

6.6.1.2.2 GTA_ERROR_ACCESS_POLICY

#define GTA_ERROR_ACCESS_POLICY 14

The access policy is invalid or cannot be used for this type of object.

6.6.1.2.3 GTA_ERROR_ATTRIBUTE_MISSING

#define GTA_ERROR_ATTRIBUTE_MISSING 13

A mandatory attribute for the function is missing.

6.6.1.2.4 GTA_ERROR_CONTEXT_BUSY

#define GTA_ERROR_CONTEXT_BUSY 16

The context cannot be used for the intended operation as another operation is still pending.

6.6.1.2.5 GTA_ERROR_ENUM_NO_MORE_ITEMS

#define GTA_ERROR_ENUM_NO_MORE_ITEMS 8

An enumeration went past the last item.

6.6.1.2.6 GTA_ERROR_FEATURE_NOT_SUPPORTED

#define GTA_ERROR_FEATURE_NOT_SUPPORTED 17

This optional feature is not supported by this implementation of GTA API.

6.6.1.2.7 GTA_ERROR_GENERIC_DEVICE_ERROR

#define GTA_ERROR_GENERIC_DEVICE_ERROR -1

Negative error codes are reserved for device specific errors within the SE provider.

6.6.1.2.8 GTA_ERROR_HANDLE_INVALID

#define GTA_ERROR_HANDLE_INVALID 2

The handle is invalid.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024

6.6.1.2.9 GTA_ERROR_HANDLES_EXAUSTED

#define GTA_ERROR_HANDLES_EXAUSTED 4

The maximum amount of handles has been exhausted.

6.6.1.2.10 GTA_ERROR_INTERNAL_ERROR

#define GTA_ERROR_INTERNAL_ERROR 1

A generic internal error has occurred.

6.6.1.2.11 GTA_ERROR_INVALID_ATTRIBUTE

#define GTA_ERROR_INVALID_ATTRIBUTE 12

The specified attribute is invalid.

6.6.1.2.12 GTA_ERROR_INVALID_PARAMETER

#define GTA_ERROR_INVALID_PARAMETER 7

Some function parameter is wrong.

6.6.1.2.13 GTA_ERROR_ITEM_NOT_FOUND

#define GTA_ERROR_ITEM_NOT_FOUND 10

An object (identifier, personality, attribute) with the specified name does not exist.

6.6.1.2.14 GTA_ERROR_MEMORY

#define GTA_ERROR_MEMORY 5

There is not enough memory to complete the function.

6.6.1.2.15 GTA_ERROR_NAME_ALREADY_EXISTS

#define GTA_ERROR_NAME_ALREADY_EXISTS 9

An attempt to create an object or attribute with an already existing name failed.

6.6.1.2.16 GTA_ERROR_PROFILE_UNSUPPORTED

#define GTA_ERROR_PROFILE_UNSUPPORTED 11

The requested profile is not supported for this operation.

6.6.1.2.17 GTA_ERROR_PROVIDER_INVALID

#define GTA_ERROR_PROVIDER_INVALID 6

The secure element provider is invalid.

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 30168:2024