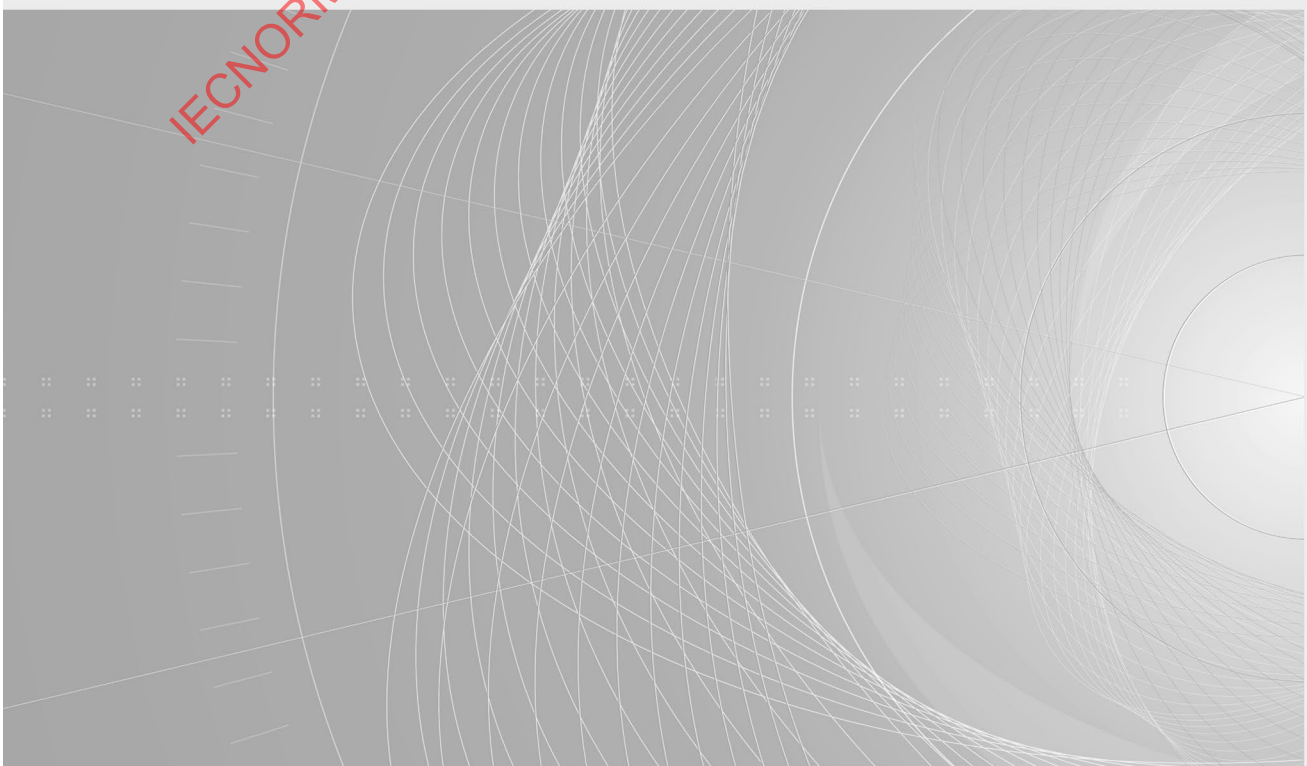
IEC 63563-9

Edition 1.0 2025-02

INTERNATIONAL STANDARD

Qi Specification version 2.0 – Part 9: Authentication Protocol

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025



IEC 63563-9:2025-02(en)

THIS PUBLICATION IS COPYRIGHT PROTECTED
Copyright © 2025 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews, graphical symbols and the glossary. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 500 terminological entries in English and French, with equivalent terms in 25 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

IEC 63563-9

Edition 1.0 2025-02

INTERNATIONAL STANDARD

Qi Specification version 2.0 – Part 9: Authentication Protocol

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 29.240.99; 35.240.99

ISBN 978-2-8327-0192-8

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

® Registered trademark of the International Electrotechnical Commission

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

QI SPECIFICATION VERSION 2.0 –

Part 9: Authentication Protocol

FOREWORD

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

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

IEC 63563-9 has been prepared by technical area 15: Wireless Power Transfer, of IEC technical committee 100: Audio, video and multimedia systems and equipment. It is an International Standard.

It is based on *Qi Specification version 2.0, Authentication Protocol* and was submitted as a Fast-Track document.

The text of this International Standard is based on the following documents:

Draft	Report on voting
100/4253/FDIS	100/4284/RVD

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

The language used for the development of this International Standard is English.

The structure and editorial rules used in this publication reflect the practice of the organization which submitted it.

This document was developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

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

- reconfirmed,
- withdrawn, or
- revised.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

WIRELESS POWER CONSORTIUM

Qi Specification

Authentication Protocol

Version 2.0

April 2023

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

DISCLAIMER

The information contained herein is believed to be accurate as of the date of publication, but is provided "as is" and may contain errors. The Wireless Power Consortium makes no warranty, express or implied, with respect to this document and its contents, including any warranty of title, ownership, merchantability, or fitness for a particular use or purpose. Neither the Wireless Power Consortium, nor any member of the Wireless Power Consortium will be liable for errors in this document or for any damages, including indirect or consequential, from use of or reliance on the accuracy of this document. For any further explanation of the contents of this document, or in case of any perceived inconsistency or ambiguity of interpretation, contact: info@wirelesspowerconsortium.com.

RELEASE HISTORY

Specification Version	Release Date	Description
v2.0	April 2023	Initial release of the v2.0 Qi Specification.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table of Contents

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

1 General

The Wireless Power Consortium (WPC) is a worldwide organization that aims to develop and promote global standards for wireless power transfer in various application areas. A first application area comprises flat-surface devices such as mobile phones and chargers in the Baseline Power Profile (up to 5 W) and Extended Power Profile (above 5 W).

1.1 Structure of the Qi Specification

General documents

- Introduction
- Glossary, Acronyms, and Symbols

System description documents

- Mechanical, Thermal, and User Interface
- Power Delivery
- Communications Physical Layer
- Communications Protocol
- Foreign Object Detection
- NFC Tag Protection
- Authentication Protocol

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

1.2 Scope

The *Qi Specification, Authentication Protocol* (this document) defines the architecture and application-level messaging for the Authentication of a Power Transmitter Product by a Power Receiver to ensure that the Power Transmitter Product is both Qi certified and the product of a registered manufacturer.

1.3 Compliance

All provisions in the *Qi Specification* are mandatory, unless specifically indicated as recommended, optional, note, example, or informative. Verbal expression of provisions in this Specification follow the rules provided in ISO/IEC Directives, Part 2.

Table 1: Verbal forms for expressions of provisions

Provision	Verbal form
requirement	“shall” or “shall not”
recommendation	“should” or “should not”
permission	“may” or “may not”
capability	“can” or “cannot”

1.4 References

For undated references, the most recently published document applies. The most recent WPC publications can be downloaded from http://www.wirelesspowerconsortium.com.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

1.5 Conventions

1.5.1 Notation of numbers

- Real numbers use the digits 0 to 9, a decimal point, and optionally an exponential part.
- Integer numbers in decimal notation use the digits 0 to 9.
- Integer numbers in hexadecimal notation use the hexadecimal digits 0 to 9 and A to F, and are prefixed by "0x" unless explicitly indicated otherwise.
- Single bit values use the words ZERO and ONE.

1.5.2 Tolerances

Unless indicated otherwise, all numeric values in the *Qi Specification* are exactly as specified and do not have any implied tolerance.

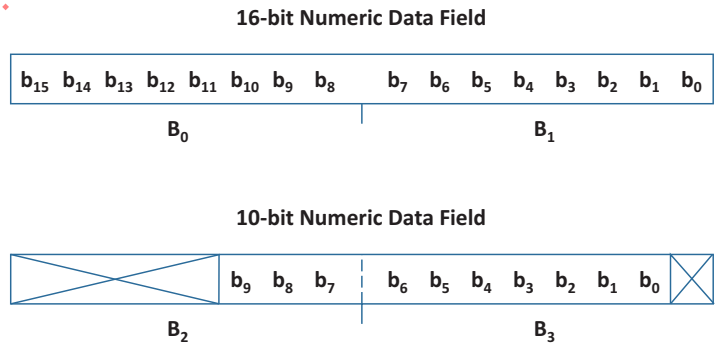
1.5.3 Fields in a data packet

A numeric value stored in a field of a data packet uses a big-endian format. Bits that are more significant are stored at a lower byte offset than bits that are less significant. Table 2 and Figure 1 provide examples of the interpretation of such fields.

Table 2: Example of fields in a data packet

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	(msb) 16-bit Numeric Data Field							
B_1								(lsb)
B_2	Other Field					(msb)		
B_3	10-bit Numeric Data Field						(lsb)	Field

Figure 1. Examples of fields in a data packet



IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

1.5.4 Notation of text strings

Text strings consist of a sequence of printable ASCII characters (i.e. in the range of 0x20 to 0x7E) enclosed in double quotes ("). Text strings are stored in fields of data structures with the first character of the string at the lowest byte offset, and are padded with ASCII NUL (0x00) characters to the end of the field where necessary.

EXAMPLE: The text string “WPC” is stored in a six-byte fields as the sequence of characters 'W', 'P', 'C', NUL, NUL, and NUL. The text string “M:4D3A” is stored in a six-byte field as the sequence 'M', ':', '4', 'D', '3', and 'A'.

1.5.5 Short-hand notation for data packets

In many instances, the *Qi Specification* refers to a data packet using the following shorthand notation:

<MNEMONIC>/<modifier>

In this notation, <MNEMONIC> refers to the data packet's mnemonic defined in the *Qi Specification, Communications Protocol*, and <modifier> refers to a particular value in a field of the data packet. The definitions of the data packets in the *Qi Specification, Communications Protocol*, list the meanings of the modifiers.

For example, EPT/cc refers to an End Power Transfer data packet having its End Power Transfer code field set to 0x01.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

1.6 Power Profiles

A Power Profile determines the level of compatibility between a Power Transmitter and a Power Receiver. Table 3 defines the available Power Profiles.

- *BPP PTx*: A Baseline Power Profile Power Transmitter.
- *EPP5 PTx*: An Extended Power Profile Power Transmitter having a restricted power transfer capability, i.e. $P_L^{(pot)}$ = 5 W.
- *EPP PTx*: An Extended Power Profile Power Transmitter.
- *BPP PRx*: A Baseline Power Profile Power Receiver.
- *EPP PRx*: An Extended Power Profile Power Receiver.

Table 3: Capabilities included in a Power Profile

Feature	BPP PTx	EPP5 PTx	EPP PTx	BPP PRx	EPP PRx
A*x* or B*x* design	Yes	Yes	No	N/A	N/A
MP-A*x* or MP-B*x* design	No	No	Yes	N/A	N/A
Baseline Protocol	Yes	Yes	Yes	Yes	Yes
Extended Protocol	No	Yes	Yes	No	Yes
Authentication	N/A	Optional	Yes	N/A	Optional

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

2 Overview

The *Qi Specification, Authentication Protocol* (this document) defines a protocol for a Power Receiver to authenticate a Power Transmitter. In this context, Authentication is a tamper-resistant method to establish and verify the identity of the Power Transmitter, enabling the Power Receiver to trust the Power Transmitter to operate within the bounds of the *Qi Specification*. This Authentication protocol version 1.0 makes use of Data Transport Streams between the Power Receiver and Power Transmitter as defined in the *Qi Specification, Communications Protocol*.

Authentication allows an organization to set and enforce a policy with regard to acceptable products. This will permit useful security assurances in real world situations. For example, a mobile phone manufacturer concerned about product damage or safety hazards resulting from substandard wireless charging devices can set a policy limiting the power drawn from an untrusted wireless charger.

This document aims to be closely aligned with the USB Authentication specification, particularly as it is likely that products will exist in the market that support both.

In addition to this document, the *WPC Manufacturer Agreement* covers legal and implementation requirements, including secure storage and handling of secrets (Annex A) and revocation rules and procedure (Annex B). For further information or to obtain a copy of this agreement, contact: info@wirelesspowerconsortium.com.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

2.1 References

Unless specified otherwise, all standards specified, including those from ISO, ITU, and NIST refer to the version or edition which is more recent, as of 1 January 2018.

ECDSA

- ANSI X9.62-2005; Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) (available at www.global.ihs.com or https://www.techstreet.com)
- NIST-FIPS-186-4, Digital Signature Standard (DSS), Section 6, Federal Information Processing Standards Publication, July 2013 (available at: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf)
- ISO/IEC 14888-3 Digital signatures with appendix—Part 3: Discrete logarithm based mechanisms (Clause 6.6)

NIST P-256, secp-256r1

- NIST-FIPS-186-4, Digital Signature Standard (DSS), Appendix D: Recommended Elliptic Curves for Federal Government Use, Federal Information Processing Standards Publication, July 2013 (available at: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf)
- ISO/IEC 15946 Cryptographic techniques based on elliptic curves (NIST P-256 is included as example)

  NOTE: The ISO/IEC 15946 series treats elliptic curves differently from FIPS 186- 4. ISO/IEC 15946-5 is about elliptic curve generation. That is, based on the method in part 5, each application and implementation can generate its own curves to use. In other words, there are no ISO/IEC recommended curves. P-256 is considered an example in ISO/IEC 15946. In addition, Elliptic Curve signatures and key establishment schemes have been moved to ISO/IEC 14888 and ISO/IEC 11770 respectively, together with other discrete-log based mechanisms. Test vectors (examples) using P-256 are included for each of those mechanisms.

SEC 1

- Certicom Corp., Standards for Efficient Cryptography Group (SECG), SEC 1: “Elliptic Curve Cryptography,” Version 1.0, September 2000 (available at: https://www.secg.org/SEC1-Ver-1.0.pdf)

SEC 2

- Certicom Corp., Standards for Efficient Cryptography Group (SECG), SEC 2: “Recommended Elliptic Curve Domain Parameters,” Version 2.0, January 2010 (available at: http://www.secg.org/sec2-v2.pdf)

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

SHA-256

- NIST-FIPS-180-4 (available at: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf)
- ISO/IEC 10118-3 Hash-functions—Part 3: Dedicated hash-functions (Clause 10)

SP800-90A

- NIST-SP-800-90A Revision 1 (available at: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf.)

SP800-90B

- NIST-SP-800-90B (available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf)

USB Authentication

- Universal Serial Bus Type-C™ Authentication Specification (available at http://www.usb.org/developers/docs/ as part of the USB 3.2 Specification download package)

X.509

- ITU-T-X.509, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute Certificate frameworks (available at: https://www.itu.int/rec/T-REC-X.509/en)
- ISO/IEC 9594-8, Information technology—Open Systems Interconnection—The Directory—Part 8: The Directory: Public-key and attribute Certificate frameworks (available at: https://www.iso.org/standard/80325.html)
- IETF-RFC-5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

2.2 Cryptographic methods

This specification targets a 128-bit security level for all cryptographic primitives. The cryptographic methods used by this specification are shown in Table 5 (in Section 3.1).

2.2.1 Random number generators

The generation of cryptographic keys and the cryptographic protocol exchanges rely on cryptographic quality random numbers. Random numbers are defined as numbers that are distinguishable from random by no algorithm with an algorithmic complexity of less than $O(2^{128})$.

The output of a NIST SP800-90A-compliant PRNG seeded with a 256-bit full SP800-90B entropy value is sufficient to meet this standard.

2.3 Security overview

Security of the Authentication protocol depends on the protection of private keys, and that protection is supported by security evaluation.

2.3.1 Methodology

This specification defines a Certificate-based method for Authentication that allows a Power Receiver to authenticate a Power Transmitter and, by policy, choose how to interact with that Power Transmitter. For example, a Power Receiver may choose not to use the full advertised capabilities of an unauthenticated Power Transmitter.

2.3.2 Periodic re-Authentication

The Power Receiver can optionally perform periodic re-Authentications to verify that an authenticated Power Transmitter has not been replaced by a different one.

2.4 Impact to existing ecosystem

The impact to existing Power Transmitter Products depends largely on individual Policy decisions regarding legacy Power Transmitters (i.e. Power Transmitters that predate this specification). For example, a Power Receiver with a Policy that allows full functioning of legacy Power Transmitters will have a minimal impact on the current ecosystem, while a Power Receiver with a Policy that limits or refuses to use functionalities exposed by a legacy Power Transmitter will have a more significant impact.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

2.5 Support for revocation

If the Power Receiver supports Authentication functionality, then the Power Receiver should support Revocation.

Revocation Lists are normally updated off line, e.g. as part of a software or firmware update or during Product manufacturing. Manufacturers are not required to provide on-line maintenance of Revocation Lists or to make Revocation Lists field-updatable in their Products, however, the Power Receiver should frequently update its revocation list and not only during firmware updates. For some Power Receiver Products with Internet connectivity (e.g. smartphones), frequent Revocation List updates should be relatively easy. Other Power Receiver Products without direct Internet connectivity, such as fitness trackers, may not be able to update their Revocation List as frequently, but manufacturers should consider ways to keep the Revocation List current in their product designs.

NOTE: Support for updatable Revocation Lists implies field-updatable non-volatile memory.

NOTE: The WPC Authentication License Administrator (WPC-ALA) will issue Revocation Lists from time to time. The format of Revocation Lists and how they are communicated is outside the scope of this specification.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3 Certificates and private keys

3.1 Certificate Chains

The requirements in this section apply to Certificate Chains that are populated in slots 0 and 1. The Power Transmitter Product manufacturer determines the requirements for Certificates that are populated in slots 2 and 3. For further information about slots, see Section 3.3, *Certificate Chain slots*.

A Certificate Chain consists of three Certificates; see Table 4. The first Certificate in the chain is the Root Certificate identifying the WPC Root Certificate Authority. It is a self-signed Certificate (i.e. signed by the WPC Root Certificate Authority). As an optimization, the Certificate Chain contains a hash of the Root Certificate rather than the Root Certificate itself. The second Certificate is a Manufacturer CA Certificate identifying the product's manufacturer. It is signed by the WPC Root Certificate Authority. The last Certificate in the chain is a Product Unit Certificate identifying the individual Power Transmitter product. It is signed by the Manufacturer Certificate Authority.

In addition to identification data, the Manufacturer CA Certificate contains a public key for verifying the authenticity of the Product Unit Certificate. The Product Unit Certificate contains a public key for verifying the authenticity of the Power Transmitter Product Unit. See Section 4, *Authentication protocol*, for details.

The following requirements apply to the Certificates in a Certificate Chain.

- Each Product Unit Certificate shall identify a unique Power Transmitter Product Unit by using, for example, the product unit's RSID number (wpc-qi-rsid).
- Each Power Transmitter Product Unit shall have a unique public key.
- Product Unit Certificates contained in different slots of the same Power Transmitter Product Unit may contain the same identity and public key, provided that this does not introduce a security vulnerability.
- A manufacturer may have multiple Manufacturer CA Certificates to identify, for example, different production facilities or product families. Each of these Manufacturer CA Certificates shall contain a unique public key relating to a unique private key.

  NOTE: Using multiple Manufacturer CA Certificates is encouraged to partition volumes of product units to limit the collateral impact caused in the event that a Manufacturer CA Certificate is revoked (collateral impact of revoking all product units signed by that Manufacturer CA Certificate).
- A manufacturer shall use the private key associated with a Manufacturer CA Certificate to sign Product Unit Certificates.
- The WPC Root Certificate Authority shall ensure that all Manufacturer CA Certificates contain unique public keys.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

The Certificate Chains populated in slots 0 and 1 shall use X.509 Certificates. See Section 3.3, *Certificate Chain slots*, for further information about slots 0 and 1.

Table 4: Certificate Chain (X.509 Certificates)

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0...B_1	msb	Length						
								lsb
B_2...$B_{(1+N\_RH)}$	Root Certificate Hash (length N_RH)							
$B_{(2+N\_RH)}$...$B_{(1+N\_RH+N\_MC)}$	Manufacturer CA Certificate (length N_MC)							
$B_{(2+N\_RH+N\_MC)}$... $B_{(1+N\_RH+N\_MC+N\_PUC)}$	Product Unit Certificate (length N_PUC)							
Note: The Root Certificate is not included in the chain, only its hash is included in the chain.								

Length The length is the total number of bytes in the Certificate Chain including the Length field.

Root Certificate Hash A SHA-256 hash of the Root Certificate. The length of N_RH is 32 bytes. See Section 2, *Overview*.

Manufacturer CA Certificate This Certificate identifies the manufacturer.

Product Unit Certificate This Certificate identifies the individual product unit.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3.2 Certificates

3.2.1 Format of Certificates

All Certificates shall use:

- the X.509 v3 ASN.1 structure,
- binary DER encoding for ASN.1, and
- the cryptographic methods listed in Table 5.

Table 5: Summary of cryptographic methods

Method	Use
X.509 v3, DER encoding	Certificate format
ECDSA using the NIST P256, secp256r1 curve, uncompressed point OR compressed point format as specified by SEC1 (see Section 2, *Overview*)	Digital signing of Certificates and Authentication Messages
SHA-256	Hash algorithm, used in the ECDSA calculation and in creating digests of Certificates and Certificate Chains.

The further description of the Certificate format assumes that the reader is familiar with X.509 v3 Certificate terminology.

NOTE: Certificates and the fields, attributes, and extensions defined therein are Big Endian.

Product unit Certificates shall not exceed MaxProdCertSize in length. A Manufacturer CA Certificate shall not exceed MaxManufacturerCertSize in length.

3.2.1.1 Textual format

All textual ASN.1 objects contained within X.509 Certificates, including DirectoryString, GeneralName, and DisplayText, shall be specified as a UTF8String. The length of any textual object shall not exceed 64 bytes excluding the DER type and DER length encoding.

3.2.1.2 Attributes and Extensions

Where applicable, the Object Identifier (OID) is provided.

Basic Constraints (OID 2.5.29.19)

Manufacturer CA Certificates contain this extension. (The extension is marked as critical, its cA component is set to true, and its pathLenConstraint component set to zero). Product Unit Certificates do not contain this extension.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Validity

The notBefore and notAfter fields indicate the time interval during which information regarding a Certificate's validity is maintained. For Product Units, the validity times should be ignored.

Certificate notBefore and notAfter validity times shall be specified using ASN.1 GeneralizedTime for any year, or ASN.1 UTCTime for years prior to 2050.

For a Manufacturer CA Certificate it is recommended that the notBefore field be "19700101000000Z" (for 00:00 on 01-Jan-1970 UTC, which is POSIX epoch time). It is recommended that the notAfter field be "99991231235959Z" (for 23:59:59 on 31-Dec-9999 UTC, which is used for an unknown expiration time as defined in IETF-RFC-5280, Section 4.1.2.5). Use of the recommended notBefore and notAfter values will maximize compatibility with Certificate processing stacks.

Qi policy extension (wpc-qi-policy)

The WPC Qi policy extension is a custom Manufacturer CA Certificate extension for use with products compliant to this specification. The value is a binary object and is reserved in this version of the specification. The value of wpc-qi-policy is listed on The Qi Specifications page of the WPC members' website.

The binary object is encoded as an ASN.1 DER OCTET STRING with a fixed size of QiPolicySize bytes.

Manufacturer CA Certificates shall contain this extension. Product Unit Certificates shall not contain this extension.

Qi RSID extension (wpc-qi-rsid)

The WPC Qi revocation sequential identifier (RSID) extension is a custom Product Unit Certificate extension for use with products compliant to this specification. The value is a binary object and is provided to store a sequential identifier unique to each product unit that enables range-based revocation of batches. The value of wpc-qi-rsid is listed on The Qi Specifications page of the WPC members' website.

The binary object is DER encoded as an ASN.1 OCTET STRING with a maximum size of MaxQiRSIDSize bytes. The RSID value is unique to each product unit, from a minimum of 1 up to MaxQiRSIDSize bytes.

Product Unit Certificates shall contain this extension. Manufacturer CA Certificates shall not contain this extension.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

ASN.1 schema for WPC Object identifiers

The ASN.1 module below defines the WPC administered object identifiers and the schema for the certificate extensions used for WPC Qi authentication.

```
-- WPC OID allocation

wpc INTEGER ::= 148

id-wpc OBJECT IDENTIFIER

  ::= {joint-iso-ccitt(2) international-organizations(23) wpc}

-- Qi Authentication certificate extensions

qiAuth INTEGER ::= 1

id-wpc-qiAuth OBJECT IDENTIFIER ::= {id-wpc qiAuth}

policy INTEGER ::= 1

id-wpc-qiAuth-policy OBJECT IDENTIFIER ::= {id-wpc-qiAuth policy}

rsid INTEGER ::= 2

id-wpc-qiAuth-rsid OBJECT IDENTIFIER ::= {id-wpc-qiAuth rsid}

-- WPC Qi Authentication size constants

wpcQiAuthPolicySize INTEGER ::= 4

wpcQiAuthRsidMaxSize INTEGER ::= 9

-- WPC Qi Authentication certificate extension schema

policy ::= OCTET STRING (SIZE (wpcQiAuthPolicySize))

rsid ::= OCTET STRING (SIZE (1…wpcQiAuthRsidMaxSize))
```

3.2.1.3 Manufacturer CA Certificate

A Manufacturer CA Certificate uses the ASN.1 structure defined in Rec. ITU-T X.509 (10/2016), Section 7.2 Public-Key Certificate.

The TBSCertificate component of a Manufacturer CA Certificate only implements the ASN.1 fields shown in Table 6.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 6: Certificate profile for a Manufacturer CA Certificate

	Field	OID	Data type	Value	Example
TBSCertificate	Version		INTEGER	0x02 {=X.509v3}	0x02
	SerialNumber		INTEGER	A unique 1- to 72-bit positive integer value	0x01 10 20 30 40 50 60 70
		Note: The maximum length of 9 bytes refers to the “baseband” original value, before DER encoding. The encoded value can have an additional zero byte if the top bit of the original value is set. Be aware that this can be 10 bytes after encoding.			
	signature	1.2.840.10045.4.3.2 {ecdsa-with-SHA256}		N/A	
	issuer	2.5.4.3 {Common Name}	UTF8String	“WPCCA“+ one character suffix denoting different root CA instances	"WPCCA1"
	validity.notBefore		Either Generalized Time for any year, or UTCTime for years prior to 2050	Any value (not used by PRx)	2000-01-01 00:00:00
	validity.notAfter		Either Generalized Time for any year, or UTCTime for years prior to 2050	Any value (not used by PRx)	9999-12-31 23:59:59
	subject	2.5.4.3 {Common Name}	UTF8String	7 byte string consisting of: ▪ four upper-case characters containing a PTMC hex value ▪ one character containing a dash ▪ two arbitrary alpha-numeric characters	“CACA-1A”

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 6: Certificate profile for a Manufacturer CA Certificate (Continued)

	Field	OID	Data type	Value	Example
TBSCertificate	subjectPublicKeyInfo.algorithm	1.2.840.10045.2.1 {ecPublicKey}		N/A	
		1.2.840.10045.3.1.7* {secp256r1}		N/A	
	subjectPublicKeyInfo.subjectPublicKey		BIT STRING	(Public Key value; optionally may use compressed point representation)†	0x04:64:68:34:24:4e:1a:37:b2:f8:3a:d5:30:68:73:8a:65:9b:e2:a6:d2:c6:f3:c9:3f:90:5e:8c:7d:a3:59:79:27:6b:a7:fb:d4:30:83:42:a9:90:a7:c1:92:90:98:20:7d:90:77:f2:97:f3:f5:3a:77:25:01:3c:55:44:19:e8:4a
	Extensions.1	2.5.29.19 {basicConstraints}		N/A	
	Extensions.1.critical		BOOLEAN	TRUE	TRUE
	Extensions.1.extnValue.cA		BOOLEAN	TRUE	TRUE
	Extensions.1.extnValue.pathLenConstraint		INTEGER	0	0x00
	Extensions.2	2.23.148.1.1 {wpc-qiAuth-policy}		N/A	
	Extensions.2.critical		BOOLEAN	TRUE	TRUE
	Extensions.2.extnValue		OCTET STRING	4 octets (bytes) reserved for future use	0x00 00 00 01

* 1.2.840.10045 refers to the named curve defined equivalently by three organizations as: NIST (FIPS186-4): "P-256", SECG (SEC 2): "secp256r1", and IETF (RFC 3279): "prime256v1"

† A Certificate verifier can unambiguously distinguish compressed from uncompressed point representations by evaluating the value of the first byte (0x02 and 0x03 for compressed points and 0x04 for an uncompressed point).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3.2.1.4 Product Unit Certificate

A Product Unit Certificate uses the ASN.1 structure defined in Rec. ITU-T X.509 (10/2016), Section 7.2 Public-Key Certificate.

The TBSCertificate component of a Product Unit Certificate only implements the ASN.1 fields shown in Table 7.

Table 7: Certificate profile for a Product Unit Certificate

	Field	OID	Data Type	Value	Example
TBSCertificate	Version		INTEGER	0x02 {=X.509v3}	0x02
	serialNumber		INTEGER	A unique 1- to 72-bit positive integer value	0x01 10 20 30 40 50 60 70
		Note: The maximum length of 9 bytes refers to the "baseband" original value, before DER encoding. The encoded value can have an additional zero byte if the top bit of the original value is set. Be aware that this can be 10 bytes after encoding.			
	signature	1.2.840.10045.4.3.2 {ecdsa-with-SHA256}		N/A	
	issuer	2.5.4.3 {Common Name}	UTF8String	7 bytes string consisting of: ▪ Four upper-case characters containing a PTMC hex value ▪ one character containing a dash ▪ two arbitrary alpha-numeric characters ▪ this matches exactly byte-for-byte the value from the Manufacturer CA Certificate Subject common name	"CACA-1A"
	validity.notBefore		Either Generalized Time for any year, or UTCTime for years prior to 2050	Should use the current time when issuing the Certificate (not used by PRx)	2020-10-01 00:00:00

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 7: Certificate profile for a Product Unit Certificate (Continued)

	Field	OID	Data Type	Value	Example
TBSCertificate	validity.notAfter		Either Generalized Time for any year, or UTCTime for years prior to 2050	Should use one day after NotBefore (not used by PRx)	2020-10-02 00:00:00
	subject.attribute1	2.5.4.3 {Common Name}	UTF8String	String consisting of: ▪ The Qi ID encoded as a six-character text string, left-padded with UTF-8 “0” (zero) characters as necessary. For example, the Subject ID for a product with Qi ID 6386 is encoded as “006386“ ▪ Optionally one character containing a dash followed by up to 28 arbitrary characters to improve readability e.g. product name	"006386-Model5"
	subject.attribute2 {optional}	2.5.4.92 {tagAFI}	OCTET STRING	Optional: up to 32 arbitrary bytes	0x20 20 09 23 F4
	subject.attribute3 {optional}	0.9.2342.19200300.100.1.1 {userId}	UTF8String	Optional: up to 32 arbitrary characters	“SEPT-23“
	subjectPublicKeyInfo.algorithm	1.2.840.10045.2.1 {ecPublicKey}		N/A	
		1.2.840.10045.3.1.7* {secp256r1}		N/A	

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 7: Certificate profile for a Product Unit Certificate (Continued)

	Field	OID	Data Type	Value	Example
TBSCertificate	subjectPublicKeyInfo.subjectPublicKey		BIT STRING	(Public Key value; optionally may use compressed point representation)†	0x04:64:68:34:24:4e:1a:37:b2:f8:3a:d5:30:68:73:8a:65:9b:e2:a6:d2:c6:f3:c9:3f:90:5e:8c:7d:a3:59:79:27:6b:a7:fb:d4:30:83:42:a9:90:a7:c1:92:90:98:20:7d:90:77:f2:97:f3:f5:3a:77:25:01:3c:55:44:19:e8:4a
	Extensions.1	2.23.148.1.2 {wpc-qiAuth-rsid}		N/A	
	Extensions.1.critical		BOOLEAN	TRUE	TRUE
	Extensions.1.extnValue		OCTET STRING	RSID	0x00 00 00 00 00 00 00 00 01

* 1.2.840.10045 refers to the named curve defined equivalently by three organizations as:
NIST (FIPS186-4): "P-256"
SECG (SEC 2): "secp256r1"
IETF (RFC 3279): "prime256v1"

† A Certificate verifier can unambiguously distinguish compressed from uncompressed point representations by evaluating the value of the first byte (0x02 and 0x03 for compressed points and 0x04 for an uncompressed point).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3.2.1.5 Root CA Certificate

A Root CA Certificate uses the ASN.1 structure defined in Rec. ITU-T X.509 (10/2016), Section 7.2 Public-Key Certificate.

The TBSCertificate component of a Root Certificate only implements the ASN.1 fields shown in Table 8.

Table 8: Certificate profile for a Root CA Certificate

	Field	OID	Data Type	Value	Example
TBSCertificate	Version		INTEGER	0x02 {=X.509v3}	0x02
	serialNumber		INTEGER	A unique number up to 9 bytes in length	0x01 10 20 30 40 50 60 70
	signature	1.2.840.10045.4.3.2 {ecdsa-with-SHA256}		N/A	
	issuer	2.5.4.3 {Common Name}	UTF8String	"WPCCA" + one character suffix denoting different root CA instances	"WPCCA1"
	validity.notBefore		Either Generalized Time for any year, or UTCTime for years prior to 2050	Any value (not used by PRx)	2000-01-01 00:00:00
	validity.notAfter		Either Generalized Time for any year, or UTCTime for years prior to 2050	Any value (not used by PRx)	9999-12-31 23:59:59
	subject	2.5.4.3 {Common Name}	UTF8String	Same as "issuer"	"WPCCA1"

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 8: Certificate profile for a Root CA Certificate (Continued)

	Field	OID	Data Type	Value	Example
TBSCertificate	subjectPublicKeyInfo.algorithm	1.2.840.10045.2.1 {ecPublicKey}		N/A	
		1.2.840.10045.3.1.7 {secp256r1}		N/A	
	subjectPublicKeyInfo.subjectPublicKey		BIT STRING	(Public Key value; optionally may use compressed point representation)*	0x04:64:68:34:24:4e:1a:37:b2:f8:3a:d5:30:68:73:8a:65:9b:e2:a6:d2:c6:f3:c9:3f:90:5e:8c:7d:a3:59:79:27:6b:a7:fb:d4:30:83:42:a9:90:a7:c1:92:90:98:20:7d:90:77:f2:97:f3:f5:3a:77:25:01:3c:55:44:19:e8:4a
	Extensions.1	2.5.29.19 {basicConstraints}		N/A	
	Extensions.1.critical		BOOLEAN	TRUE	TRUE
	Extensions.1.extnValue.cA		BOOLEAN	TRUE	TRUE
	Extensions.1.extnValue.pathLenConstraint	not present	N/A	N/A	N/A

* A Certificate verifier can unambiguously distinguish compressed from uncompressed point representations by evaluating the value of the first byte (0x02 and 0x03 for compressed points and 0x04 for an uncompressed point).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3.2.1.6 Additional attributes and extensions

Additional Certificate attributes and extensions defined in X.509 v3 are not allowed in a Manufacturer CA Certificate or in a Product Unit CA Certificate.

3.2.1.7 Constraints

QiPolicySize = 0x4 bytes
MaxQiRSIDSize = 0x9 bytes
MaxManufacturerCertSize = 0x200 bytes
MaxProdCertSize = 0x200 bytes

3.3 Certificate Chain slots

Certificate Chains reside in positions called slots. Each slot shall either be empty or contain one complete Certificate Chain. A Power Transmitter shall not contain more than 4 slots. Slot 0 and slot 1 (optional) shall only be used for Certificate Chains rooted with a WPC Root Certificate and shall not contain any other Certificate Chains. The manufacturer may use Slots 2 and 3 for any additional Certificate Chains. If used, slots 2 and 3 shall be populated in order starting with slot 2.

Table 9: Certificate Chain slots

Slot #	Usage
0	Mandatory; use only for Certificate Chain rooted with a WPC Root Certificate
1	Optional; use only for Certificate Chain rooted with a WPC Root Certificate
2	Optional; additional Certificate Chain if needed by the manufacturer
3	Optional; additional Certificate Chain if needed by the manufacturer

Manufacturer-dependent Certificate Chains do not need to be structured as defined in Table 4. For example, they may be in an X.509v3 format with ASN.1 DER encoding, and/or they may be signed in a manufacturer-dependent manner. Manufacturer-dependent Certificate Chains that share a public/private key pair with a Certificate Chain in one or more of slots 0 and 1 shall be constructed to provide at least the same level of security to the public/private key pair as is required for the Certificate Chains stored in slots 0 and 1. Protection for manufacturer-dependent Certificate Chains and the formats and protocol they use shall match or exceed the protection for Certificate Chains, formats, and protocols defined in the *Qi Specification, Authentication Protocol* (this document) and stored in slots 0 and 1.

3.3.1 Provisioning

Provisioning is the process by which a Power Transmitter private key acquires one or more Certificate Chains. This procedure is outside the scope of the *Qi Specification*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

3.4 Power Transmitter private keys

Each Certificate Chain in a Power Transmitter corresponds to a Power Transmitter private key whose corresponding public key is certified in the Product Unit Certificate of the slot associated with the Certificate Chain. The Power Transmitter shall have access to that private key. The Power Transmitter shall store private keys in a manner that adequately protects the confidentiality of the key as outlined in Annex A of the *WPC Manufacturer Agreement*.

A manufacturer shall not use a private key associated with one Power Transmitter Product Unit in any other Power Transmitter Product Unit or Manufacturer CA Certificate.

3.5 Other private keys

Each Manufacturer CA Certificate is associated with a unique public/private key pair. The private key associated with a Manufacturer CA Certificate is used for signing the next Certificate in a Certificate Chain. Manufacturers shall generate, provision, and store private keys in a manner that adequately protects the confidentiality of the key. See Section 2.4, *Impact to existing ecosystem* for detailed recommendations.

A Manufacturer shall not use a private key associated with one Manufacturer CA Certificate in any other Manufacturer CA Certificate or Product Unit Certificate.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

4 Authentication protocol

The Power Receiver can perform three operations:

- Query a Power Transmitter for Certificate Chain digests
- Read a Certificate Chain from a Power Transmitter
- Challenge a Power Transmitter in order to verify its authenticity

A Power Receiver may initiate as many or as few of these operations as are needed to achieve the desired Authentication latency. In addition, a Power Receiver may initiate the operations in any order. For example, a Power Receiver may first challenge a Power Transmitter, and then initiate a Certificate Chain read if the essential information from a target Certificate Chain is not already cached. Chapter 7, *Protocol flow examples*, provides some flow examples.

4.1 Digest query

To query a Power Transmitter for Certificate Chain digests, a Power Receiver sends a GET_DIGESTS request as defined in Section 5.2.1, *GET_DIGESTS*. If an error condition is encountered, the Power Transmitter shall respond with the appropriate ERROR response as defined in Section 5.3.4, *Error*. Otherwise, the Power Transmitter shall respond with a DIGESTS response as defined in Section 5.3.1, *DIGESTS*. After receiving a DIGESTS response, the Power Receiver can check to see if it has any of the Power Transmitter's Certificate Chains cached. This allows the Power Receiver to potentially skip reading a Certificate Chain and thus save time.

4.2 Certificate Chain read

To read a Certificate Chain, or portion thereof, a Power Receiver shall send a GET_CERTIFICATE request as defined in Section 5.2.2, *GET_CERTIFICATE*.

If a Power Transmitter receives a GET_CERTIFICATE request that targets an offset that is outside the Certificate Chain (i.e. offset > length) or attempts to read beyond the length of the target Certificate Chain (i.e. (offset + length) > Certificate Chain length), then the Power Transmitter shall return an ERROR response of INVALID_REQUEST.

The Power Transmitter shall respond to error conditions with the appropriate ERROR response as defined in Section 5.3.4, *Error*. Otherwise, the Power Transmitter shall respond with a CERTIFICATE response as described in Section 5.3.2, *CERTIFICATE*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

4.3 Authentication challenge

To challenge a Power Transmitter, the Power Receiver sends a CHALLENGE request as defined in Section 5.2.3, *CHALLENGE*. If an error condition is encountered, the Power Transmitter shall respond with the appropriate ERROR response as defined in Section 5.3.4, *Error*. Otherwise, the Power Transmitter shall respond with a CHALLENGE_AUTH response as described in Section 5.3.3, *CHALLENGE_AUTH*.

NOTE: A Power Receiver that supports Authentication Protocol Versions greater than 1 should ensure that the version of the CHALLENGE request and the version of the response match. The purpose of this is to resist a downgrade attack.

4.4 Errors and alerts

4.4.1 Invalid request

If a Power Transmitter receives an Authentication request with one or more invalid fields, it shall respond to that Authentication request with an ERROR response of INVALID_REQUEST as described in Section 5.3.4, *Error*.

4.4.2 Unsupported protocol version

If a Power Transmitter receives an Authentication request that contains an unsupported Authentication Protocol Version in the Qi Authentication Protocol Version field, it shall respond to that Authentication request with an ERROR response of UNSUPPORTED_PROTOCOL that has the Authentication Protocol Version set to the maximum Authentication Protocol Version it supports and the Error Data field also set to the maximum Authentication Protocol Version it supports as described in Section 5.3.4, *Error*.

4.4.3 Busy

If a Power Transmitter receives an Authentication request but is unable to meet the timing requirements listed in Chapter 6, *Timing requirements*, it shall respond to that Authentication request with an ERROR response of BUSY as described in Section 5.3.4, *Error*, within the required response time.

4.4.4 Unspecified

If a Power Transmitter, upon receiving an Authentication request, encounters an error that is not covered by the conditions above, it shall respond to that Authentication request with an ERROR response of UNSPECIFIED as described in Section 5.3.4, *Error*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5 Authentication messages

Authentication messages are used to convey information related to Authentication. Neither a Power Receiver nor a Power Transmitter shall add any padding after an Authentication message, i.e. the first byte of the Authentication message shall be the first byte of the data transport stream and the last byte of the Authentication message shall be the last byte of the data transport stream.

There are two types of Authentication messages: Authentication requests and Authentication responses. Authentication requests are defined in Section 5.2, *Authentication requests*. Authentication responses are defined in Section 5.3, *Authentication responses*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5.1 Authentication message header

All Authentication messages start with a one-byte header, which shall have the format shown in Table 10. The header may be followed by a payload, as defined in Section 5.2, *Authentication requests*, and Section 5.3, *Authentication responses*.

Table 10: Authentication message header

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Protocol Version				Message Type			

Authentication Protocol Version This field identifies which version of the Authentication Specification is being used. Table 11 shows the valid values for this field. A Product shall not use an Authentication Protocol Version value corresponding to a specification revision that it does not support.

Table 11: Authentication Protocol Version 1.0

Name	Value	Meaning
Reserved	0x0	Reserved value
V1.0	0x1	Authentication Protocol Version 1.0
Reserved	0x2 - 0xF	Reserved value

The Power Receiver shall indicate the maximum protocol version that it supports in the Authentication Protocol Version field of the first Authentication request that it sends to the Power Transmitter on a new connection. If the Power Receiver receives an ERROR response of UNSUPPORTED_PROTOCOL, it shall use the version indicated in the Error Data field of the ERROR response in all subsequent Authentication requests. If a Power Transmitter receives an Authentication request that contains an Authentication Protocol Version in the Authentication Protocol Version field that is lower than the highest version it supports, it shall use the lower version in all Authentication responses.

In all cases, the format of the message shall match that specified for the version of the Authentication Protocol Version given in the Authentication Protocol Version field.

Message Type This field identifies Authentication Message type from one of the values in Table 11 (authentication requests) or Table 16 (authentication responses).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5.2 Authentication requests

Authentication requests are used by a Power Receiver to send a command to a Power Transmitter and/or to retrieve data. Authentication request types are listed in Table 12.

A Power Receiver shall not send another Authentication request until it has either received a response for or timed out the previously-sent Authentication request.

Table 12: Authentication requests

Value	Description
0x0 - 0x7	Used for Authentication responses
0x8	Reserved
0x9	GET_DIGESTS
0xA	GET_CERTIFICATE
0xB	CHALLENGE
0xC...0xF	Reserved

5.2.1 GET_DIGESTS

This request is used to retrieve Certificate Chain digests. The Power Receiver may request any number of digests at a time. The format for a GET_DIGESTS request is defined in Table 13.

Table 13: GET_DIGESTS request

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1	Reserved			Reserved	Slot Mask			

A Power Receiver shall set all reserved bits to ZERO. A Power Transmitter shall ignore all reserved bits.

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x19.

Slot Mask This field identifies the slots (see Section 3.3, *Certificate Chain slots*) in which the requested Certificate Chains are stored. b_0 is set to ONE to request the digest of the Certificate Chain stored in slot 0, b_1 is set to ONE to request the digest of the Certificate Chain stored in slot 1, etc.

NOTE: Multiple Certificate Chain digests can be requested with a single command.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5.2.2 GET_CERTIFICATE

This request is used to read a segment of a target Certificate Chain. The format for a GET_CERTIFICATE request is defined in Table 14.

Table 14: GET_CERTIFICATE request

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1	OffsetA8			LengthA8			Slot Number	
B_2	Offset70							
B_3	Length70							

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x1A.

OffsetA8, Offset70 These two fields combine to form the offset in bytes from the start of the Certificate Chain to where the read request begins. The offset value is OffsetA8 × 256 + Offset70 and is denoted as GET_CERTIFICATE/Offset. Attempting to read beyond the end of a Certificate Chain shall result in the Power Transmitter returning an ERROR response of INVALID_REQUEST.

An offset value at or above 0x600 shall be adjusted by the PTx to return data relative to the first byte of the Product Unit Certificate (i.e. as if the Product Unit Certificate begins at 0x600).

An offset value in the range 0x600-0x7FF thus means the PTx sends bytes from the Certificate Chain at an offset of:

2 + N_RH + N_MC + GET_CERTIFICATE/Offset - 0x600

The above means that (in an example implementation) the offset value is adjusted as follows:

```
if (GET_CERTIFICATE/offset >= 0x600)
   offset =  2 + N_RH + N_MC +
GET_CERTIFICATE/offset – 0x600
else
   offset = GET_CERTIFICATE/offset;
```

LengthA8, Length70 These two fields combine to form the length in bytes of the read request. The length value is LengthA8 × 256 + Length70 and is denoted as GET_CERTIFICATE/Length. Attempting to read beyond the end of a Certificate Chain shall result in the Power Transmitter returning an ERROR response of INVALID_REQUEST. A length value of 0x000 means the PTx shall "send as many remaining bytes as possible."

The above means that (in an example implementation) the length value is adjusted as follows:

```
if (length == 0)
   length = 2 + N_RH + N_MC + N_PUC – offset;
else
   length = GET_CERTIFICATE/length - offset;
```

NOTE: This formula is not valid if GET_CERTIFICATE/offset >= 0x600.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Slot Number This is the slot number in which the target Certificate Chain is stored. The value in this field shall identify one of the Certificate Chains identified as being present in the DIGESTS response message or in the CHALLENGE_AUTH response message.

5.2.3 CHALLENGE

This request is used to initiate Authentication of a Power Transmitter Product Unit. The format for a CHALLENGE request is defined in Table 15.

Table 15: CHALLENGE request

	b_7	**b_6**	**b_5**	**b_4**	**b_3**	**b_2**	**b_1**	**b_0**
B_0	Authentication Message Header							
B_1	Reserved					Reserved	Slot Number	
B_2...B_{17}	Nonce							

A Power Receiver shall set all reserved bits to ZERO. A Power Transmitter shall ignore all reserved bits.

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x1B.

Slot Number This is the Slot number in which the Certificate Chain that will be used for Authentication is stored. The value in this field shall identify one of the Certificate Chains identified as being present in the DIGESTS response message or the CHALLENGE_AUTH response message.

NOTE: A Certificate Chain is always present in Slot 0.

Nonce This is a 128-bit binary Random Number chosen by the Power Receiver. The RNG used to generate the Nonce should meet the requirements of SP800-90A and SP800-90B (see Section 2.2.1, *Random number generators*).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5.3 Authentication responses

Power Transmitters use the response types listed in Table 16 to respond to Authentication requests.

Table 16: Authentication responses

Value	Description
0x0	Reserved
0x1	DIGESTS
0x2	CERTIFICATE
0x3	CHALLENGE_AUTH
0x4...0x6	Reserved
0x7	ERROR
0x8...0xF	Used for Authentication requests

5.3.1 DIGESTS

Power Transmitters use the DIGESTS response to send Certificate Chain digests and to report which slots contain valid Certificate Chain digests. The format for a DIGESTS response is defined in Table 17.

Table 17: DIGESTS response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1	Slots Populated Mask				Slots Returned Mask			
$B_{2 + 32 \times (n - 1)}$ ⋯ $B_{33 + 32 \times (n - 1)}$	Digests Returned							

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x11.

Slots Populated Mask The bit in position b_{K+4} of this field shall be set to ONE if and only if slot number K contains a Certificate Chain for the protocol version in the Authentication Protocol Version field.

NOTE: A Certificate Chain is always present for slot 0, and therefore bit b_4 in this mask is always set to ONE.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Slots Returned Mask The bit in position b_K of this field shall be set to ONE if and only if:

- slot number K contains a Certificate Chain for the protocol version in the Authentication Protocol Version field, and
- slot number K was set to 1 in the corresponding GET_DIGEST request.

If there are no such slots, then all bits in this mask shall be set to ZERO.

The number of digests returned shall be equal to the number of bits set in this byte. The digests shall be returned in order of increasing slot number.

Digests Returned The sequence of 32-byte SHA-256 digests of the Certificate Chain(s) for which the bit(s) in the Slots Returned Mask are non-zero, in order of increasing slot number. If all bits in Slots Returned are set to zero, then Digests Returned is not present. In the byte offset in Table 17, *n* represents the n^{th} digest returned.

5.3.2 CERTIFICATE

This response is used by a Power Transmitter to send the requested segment of a Certificate Chain. The format for a CERTIFICATE response is defined in Table 18.

Table 18: CERTIFICATE response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1 … $B_{(1 + length -1)}$	Certificate Chain Segment							

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x12.

Certificate Chain segment

The Certificate Chain segment shall contain a part of the requested segment (or the entire segment) of the requested Certificate Chain starting at the Offset requested in the corresponding GET_CERTIFICATE request. The length of this Certificate Chain segment shall be less than or equal to the length (LengthA8 × 256 + Length70) requested in the corresponding GET_CERTIFICATE request, and greater than or equal to 1.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

5.3.3 CHALLENGE_AUTH

Power Transmitters use CHALLENGE_AUTH to respond to a CHALLENGE request. The format for a CHALLENGE_AUTH response is defined in Table 19.

Table 19: CHALLENGE_AUTH response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1	Maximum Authentication Protocol Version				Slots Populated Mask			
B_2	Certificate Chain Hash LSB							
B_3...B_{34}	(msb) Signature *r* value (lsb)							
B_{35}...B_{66}	(msb) Signature *s* value (lsb)							

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x13.

Maximum Authentication Protocol Version The maximum Authentication Protocol Version supported by the Power Transmitter.

Slots Populated Mask The bit in position b_K of this byte shall be set to ONE if and only if slot number K contains a Certificate Chain for the protocol version in the Authentication Protocol Version field.

A Certificate Chain is always present for slot 0, and therefore bit b_0 in this mask is always set to ONE.

Certificate Chain Hash LSB The least significant byte (LSB) of the 32-byte SHA-256 hash of the Certificate Chain for which a challenge response is being provided.

Signature *r* value This field shall contain the 256-bit *r* value of a non-deterministic ECDSA digital signature, generated from the SHA-256 hash of TBSAuth as specified in Table 20. See ANSI X9.62-2005 for details.

Signature *s* value This field shall contain the 256-bit *s* value of a non-deterministic ECDSA digital signature, generated from the SHA-256 hash of TBSAuth as specified in Table 20. See ANSI X9.62-2005 for details.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 20: TBSAuth (for signature calculation)

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Prefix							
B_1...B_{32}	Certificate Chain Hash							
B_{33}...B_{50}	Challenge Request							
B_{51}...B_{53}	Copy of B_0...B_2 fields in the CHALLENGE_AUTH response in Table 19							

Prefix The Prefix field shall contain the ASCII representation of 'A', i.e. 0x41.

Certificate Chain Hash The Certificate Chain Hash field shall hold the SHA256 hash of the Certificate Chain for which a challenge response is being provided.

Challenge Request The Challenge Request field shall contain B0 ... B17 of the CHALLENGE request message (see Table 15) to which the CHALLENGE_AUTH is a response.

5.3.4 Error

This response is used by a Power Transmitter to transmit error information. The format for an ERROR response is defined in Table 21.

Table 21: ERROR response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	Authentication Message Header							
B_1	Error Code							
B_2	Error Data							

Authentication Message Header This field identifies the authentication protocol version and message type, as defined in Section 5.1, *Authentication message header*. It shall be set to 0x17.

Error Code Defined in Table 22

Error Data Defined in Table 22

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 22: ERROR Codes and Data.

Error Code	Value	Description	Error Data
Reserved	0x00	Reserved value	Reserved
INVALID_REQUEST	0x01	One or more request fields are invalid	0x00
UNSUPPORTED_ PROTOCOL	0x02	Requested protocol version is not supported	Maximum supported version
BUSY	0x03	Power Transmitter cannot respond now but will be able to respond in the future	0x00
UNSPECIFIED	0x04	Unspecified error has occurred	0x00
Reserved	0x05...0xEF	Reserved value	Reserved
Manufacturer defined	0xF0...0xFF	Manufacturer-defined (that is, defined by the owner of the Manufacturer Code in the Manufacturer Certificate)	Manufacturer defined

Table 23 gives the recommended behavior of a Power Receiver when it receives an ERROR response to an Authentication request.

Table 23: Recommended ERROR Behavior

Error Code	Received in response to request	Recommended behavior
Reserved INVALID_REQUEST UNSPECIFIED	GET_DIGESTS	Treat as Authentication failure
	GET_CERTIFICATE	Treat as Authentication failure
	CHALLENGE	Treat as Authentication failure
UNSUPPORTED_PROTOCOL	GET_DIGESTS GET_CERTIFICATE CHALLENGE	Retry using protocol version not exceeding Maximum supported version as reported in Error Data; if already using a protocol version not exceeding Maximum supported version, then retry once, then treat as Authentication failure
BUSY	GET_DIGESTS	Retry after waiting t_{Retry}
	GET_CERTIFICATE	Retry after waiting t_{Retry}
	CHALLENGE	Retry after waiting t_{Retry}
Manufacturer defined	GET_DIGESTS GET_CERTIFICATE CHALLENGE	Manufacturer defined

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

6 Timing requirements

6.1 Power Receiver timing requirements

Table 24 shows the timeout values a Power Receiver should apply for Authentication request messages. For the data transport layer defined in the *Qi Specification, Communications Protocol*, the timings start at the begin of the final ADC/end data packet sent in the data transport stream carrying the Authentication request.

Table 24: Power Receiver wait time values

Parameter	Wait Time Value	Description
$t_{DigestTimeout}$	43 seconds	The timeout for a GET_DIGESTS Authentication request
$t_{CertTimeout}$	($N \times 0.3 + 4$) seconds, with a minimum of 5 seconds	The timeout for a GET_CERTIFICATE Authentication request, where N represents the number of bytes requested
$t_{ChallengeAuthTimeout}$	23 seconds	The timeout for a CHALLENGE Authentication request

Note: The timeout values are based on the following considerations.

- The Power Transmitter requires some time to prepare its response (see Section 6.2, *Power Transmitter timing requirements*) for the associated intervals
- The Power Transmitter uses ADT/1 data packets to send its response
- The Power Receiver sends at least one DSR data packet every 250 ms
- The Authentication response includes a 20% overhead for communications errors

If the Power Receiver does not receive an Authentication response within the timeout of the first Authentication request, it should retry that request up to five times. If a timeout still occurs after the last retry, the Power Receiver should follow its policy corresponding to the Power Transmitter not supporting Authentication functionality.

If the Power Receiver does not receive an Authentication response within the timeout of a subsequent Authentication request (i.e. not a retry of the first Authentication request), it should consider that as an error. In this case, it should not retry the subsequent Authentication request but immediately follow its policy corresponding to the Power Transmitter not supporting Authentication.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

6.2 Power Transmitter timing requirements

A Power Transmitter shall meet the timing requirements defined in Table 25. For the data transport layer defined in the Qi Specification, Communications Protocol, the timings start at the beginning of the final ADC/end data packet sent in the data transport stream carrying the Authentication request.

Table 25: Power Transmitter Timing Requirement

Parameter	Wait Time Value	Description
$t_{DigestReady}$	3 seconds	The maximum time between sending a GET_DIGESTS Authentication request and being ready to start sending a DIGESTS Authentication response
$t_{CertReady}$	3 seconds	The maximum time between sending a GET_CERTIFICATE Authentication request and being ready to start sending a CERTIFICATE Authentication response
$t_{ChallengeAuthReady}$	3 seconds	The maximum time between sending a CHALLENGE Authentication request and being ready to start sending a CHALLENGE_AUTH Authentication response

A Power Receiver should not attempt to retrieve an Authentication response by sending a DSR/poll data packet before the corresponding timing value has expired. Doing so can cause the Power transmitter to send a BUSY Authentication response. The time it takes to send the latter delays retrieval of the expected Authentication response.

NOTE: A Power Transmitter fails a timing requirement if it cannot complete sending its Authentication response within the associated Power Receiver timeout value defined in Section 6.1, *Power Receiver timing requirements*, provided the Power Receiver sends at least one DSR data packet every 250 ms.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

7 Protocol flow examples

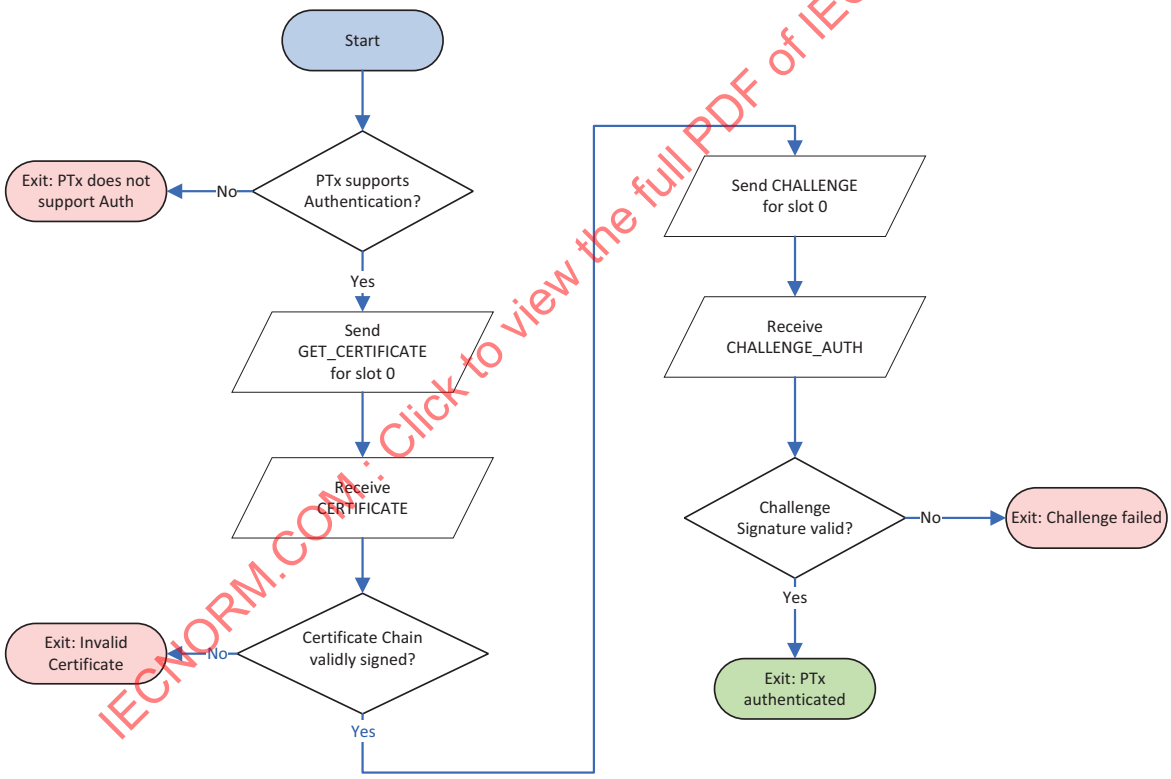
This chapter provides informative examples of suitable high level flows. It is not intended to be exclusive; other flows compliant with this specification are possible. These flows omit details such as behavior on timeouts while waiting for an Authentication response and the handling of ERROR responses. These examples also assume that the Power Receiver only processes the Certificate Chain in the Power Transmitter's slot 0. If a manufacturer intends for the Power Receiver to process Certificate Chains in other slots, the manufacturer should adapt these flows accordingly.

7.1 Simple flow

Figure 2 illustrates the behavior of a Power Receiver that neither caches the digests and public keys of previously authenticated Power Transmitters nor supports Revocation.

NOTE: This Power Receiver does not send a GET DIGESTS Authentication request because it does not cache digests.

Figure 2. Simple Power Receiver

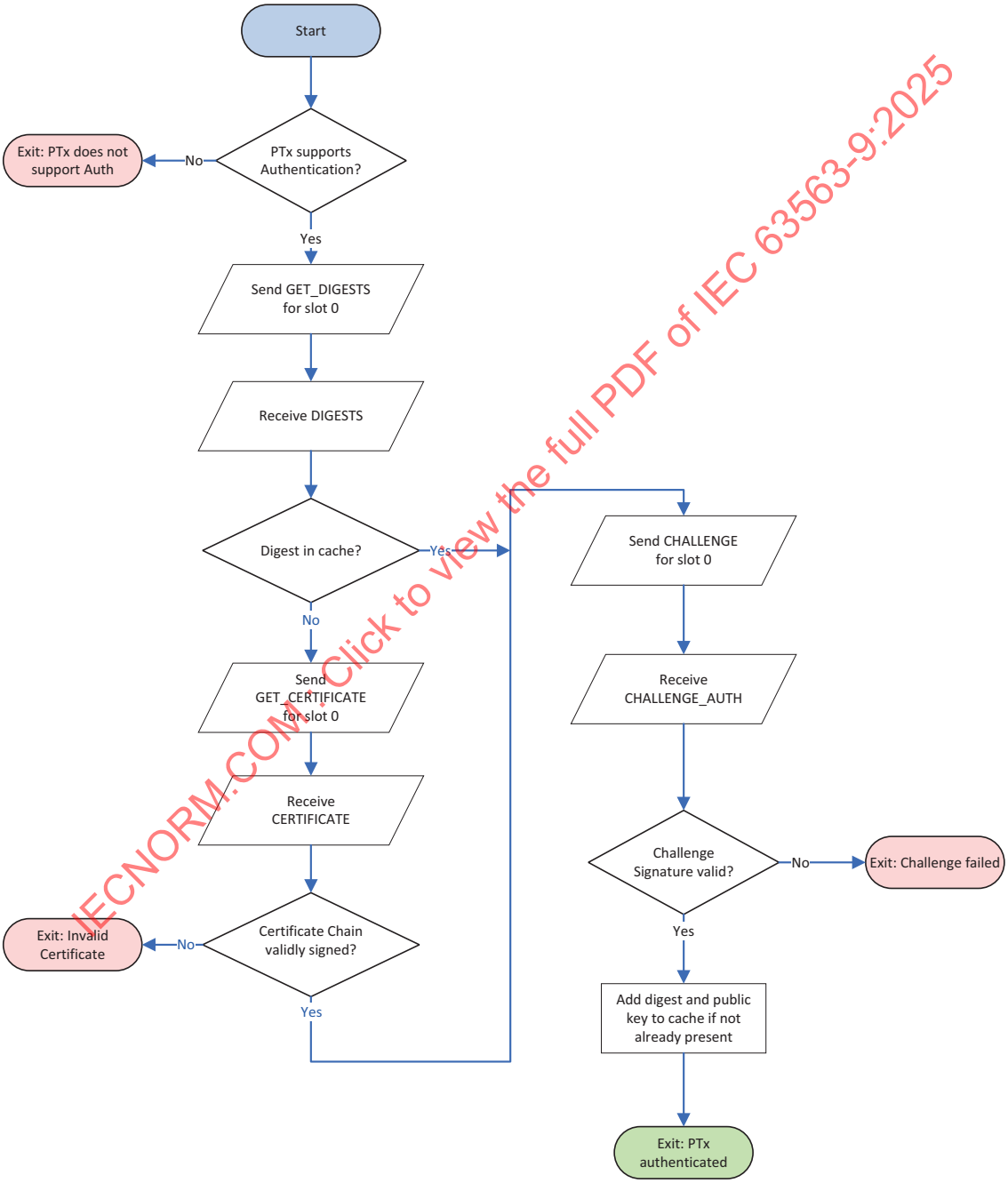


IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

7.2 Flow with caching

Figure 3 illustrates the behavior of a Power Receiver that caches (in non-volatile memory) the digests and public keys of Power Transmitters that it has previously trusted. This allows the Power Receiver to skip sending the GET CERTIFICATES Authentication request and proceed directly to issuing the CHALLENGE Authentication request. If the Certificate Chain has been copied and deployed in a counterfeit Power Transmitter, the challenge will fail even though a match will be found in the cache.

Figure 3. Power Receiver with caching

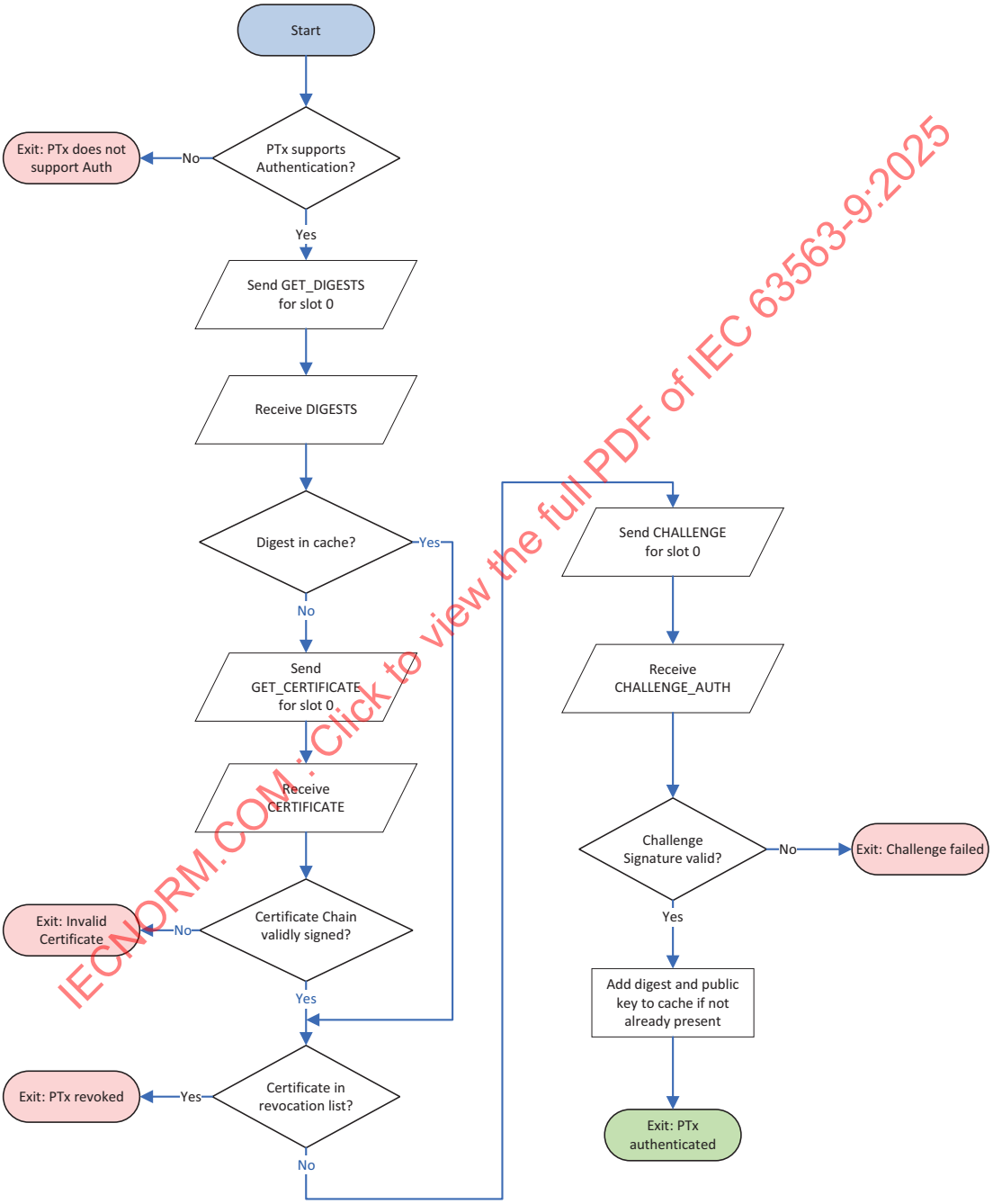


IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

7.3 Flow with caching and revocation

Figure 4 illustrates the behavior of a Power Receiver that caches (in non-volatile memory) the digests and public keys of Power Transmitters that it has previously trusted and that support Revocation. This information typically includes the Manufacturer Code, the product type, and the Product Unit Serial Number.

Figure 4. Power Receiver with caching and revocation



IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

7.4 Challenge first flow

Figure 5 illustrates an alternative behavior of a Power Receiver that caches (in non-volatile memory) the digests and public keys of Power Transmitters that it has previously trusted and that support Revocation. This behavior shares the assumptions made in Section 7.3, *Flow with caching and revocation*, on the management of the cache and Revocation Lists.

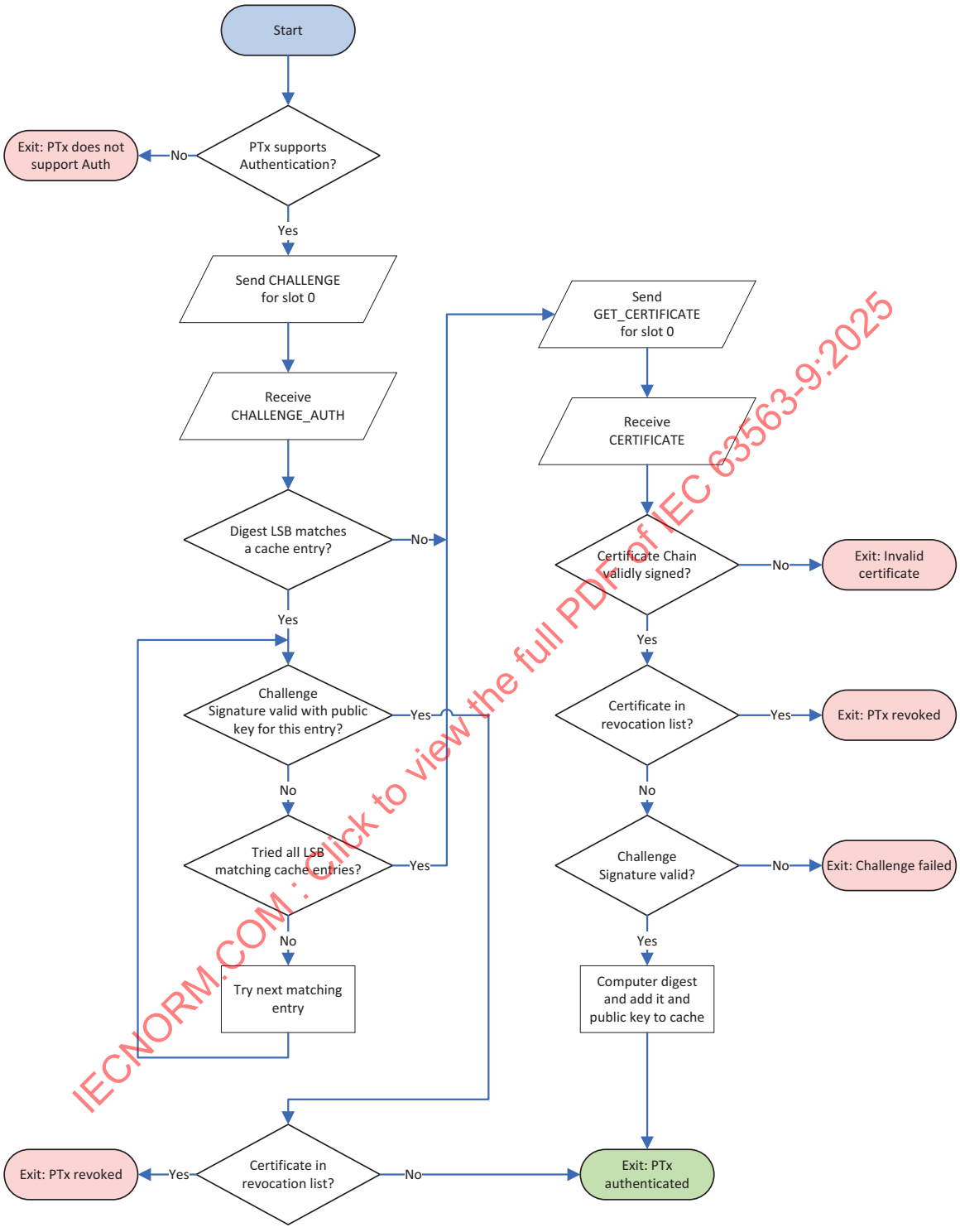
This behavior starts when the Power Receiver issues a CHALLENGE Authentication request. The CHALLENGE AUTH response contains the LSB of the hash of the Certificate Chain. The Power Receiver examines its cache of digests to try to find a digest or digests that have a matching LSB. If there are no such digests, the Power Receiver proceeds on the basis that this is a first encounter with the particular Power Transmitter. If there is a match, the Power Receiver attempts to validate the CHALLENGE AUTH signature using the public key corresponding to the matched digest. If this validation succeeds, the Power Receiver checks that the Certificate has not been revoked. If the Certificate has not been revoked, the Power Receiver determines that the responder can be trusted. If this validation fails, then the Power Receiver retries with the next digest that matches the hash LSB returned in CHALLENGE AUTH response. If all such validations fail, then the Power Receiver proceeds on the assumption that this is the first encounter with the particular Power Transmitter.

When the Power Receiver determines that this is a first encounter with the particular Power Transmitter, it retrieves the Certificate Chain, validates it, checks that it has not been revoked, and then attempts to validate the previously returned CHALLENGE AUTH signature with the public key contained in the Product Unit Certificate.

The advantage of this method is that after the first encounter between the Power Receiver and the particular Power Transmitter, only one Authentication request and response communication takes place between the two devices.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Figure 5. Challenge first Power Receiver



IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8 Cryptographic examples (informative)

The examples in this section are included to help illustrate the concepts defined in this document and are informative only.

To understand the following examples and use of them in the validation of a prototype implementation of the authentication standard, the following text includes both the values of various private keys and the value of k used in generating of each ECC signature. In any real implementation, where it is crucial that the private keys kept secret, it is very important that none of these details are exposed. As a reminder, all such values in the text are marked in either red text or with a red background.

It is important to note that the subject of revocation is completely ignored in these examples.

8.1 X.509 Certificate basics

All the certificates involved in Authentication are X.509 certificates. The ASN.1, DER-encoded X.509 certificates are created by combining the ASN.1, DER-encoded "to be signed" component of the certificate (tbsCertificate), a cryptographic signature (signatureValue) of the tbsCertificate, along with some information about the signature algorithm (signatureAlgorithm) used for creating the cryptographic signature.

The basic structure of an X.509 certificate is described by the following ASN.1 schema.

```
Certificate  := SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }
```

All Qi Authentication certificates use ECDSA with SHA256, so the AlgorithmIdentifier section is effectively constant.

The signature of an X.509 certificate can be verified by extracting the tbsCertificate portion of the ASN.1, DER-encoded certificate data (this is the 'message' that is signed), then testing the validity of the signature contained in the certificate (the signatureValue data) using the verification method associated with the particular signature algorithm used (as defined by the value of the signatureAlgorithm field of the certificate) using the public key of the "issuer CA" that the certificate claims signed the certificate.

Verification of an ECDSA signature is the process of 'decoding' the ***s*** 'proof' component of the signature data back to the original point ***R*** using the public key and the hash of the message. The test of the signature is then whether the ***r*** component of the signature matches the x-coordinate of the recovered point ***R***.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.1.1 Verifying the ECDSA signature of an X.509 Certificate

The following steps summarize the process of attempting to verify the validity of the ECDSA signature of an X.509 certificate. This process will be referred to throughout the examples chapter.

1. Extract the following data from the certificate:
   a) The raw bytes of the “tbsCertificate” portion (this is essentially the ‘message’ that is signed)
   b) The name of the “issuer” of the certificate (an identifier for the claimed ‘signatory’)
   c) The “signature” data – and de-serialize this into its (r, s) components
2. Establish the “public key” that is associated with the claimed “issuer”.
3. Check that the “signature” is a genuine ECDSA signature of the “tbsCertificate” data made with the private key associated with the claimed “public key” by using the ECDSA signature verification method (i.e. computing the SHA256 hash “digest” of the “tbsCertificate” data then checking that the ***r*** component of the “signature” is consistent with this combination of “digest”, “public key” and the ***s*** component of the “signature”).

8.2 Dummy Root CA Certificate

To expose the private key of our example manufacturer CA in these examples without the associated example Manufacturer CA Certificate being easily mistaken for a real one, the certificates presented in this chapter are all rooted in the dummy WPC Root CA Certificate described here.

8.2.1 Dummy Root CA Certificate – DER-encoded

The raw bytes of the DER-encoded, ASN.1 dummy Root CA Certificate used in these examples are shown here. A PEM-encoded version of this certificate is given in Section A:, *Sample data*.

```
30:82:01:2c:30:81:d3:a0:03:02:01:02:02:08:2f:1b:  24:bd:ee:b0:32:bd:30:0a:06:08:2a:86:48:ce:3d:04:
03:02:30:11:31:0f:30:0d:06:03:55:04:03:0c:06:57:  50:43:43:41:58:30:20:17:0d:30:30:30:31:30:31:30:
30:30:30:30:30:5a:18:0f:39:39:39:39:31:32:33:31:  32:33:35:39:35:39:5a:30:11:31:0f:30:0d:06:03:55:
04:03:0c:06:57:50:43:43:41:58:30:59:30:13:06:07:  2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:01:
07:03:42:00:04:52:aa:a0:fb:0c:4f:08:91:58:1e:0f:  c6:b6:f9:a9:3d:ae:ea:8b:95:32:eb:9d:c9:c5:ef:98:
5d:1d:80:d3:1c:8d:8e:17:7e:af:e3:da:53:8a:db:56:  d9:5d:66:90:8f:2b:43:7c:59:3c:4c:34:ed:b7:c5:a1:
ed:98:11:08:f3:a3:13:30:11:30:0f:06:03:55:1d:13:  01:01:ff:04:05:30:03:01:01:ff:30:0a:06:08:2a:86:
48:ce:3d:04:03:02:03:48:00:30:45:02:20:4c:79:ab:  6e:34:04:0b:09:4d:a3:8b:1d:6d:f4:02:73:c6:e5:b9:
c8:08:fe:3a:b0:ff:22:cc:e2:d9:b8:ab:c4:02:21:00:  a6:96:00:09:2f:55:39:6f:62:5a:9f:6e:cf:44:35:a8:
3e:4a:53:52:91:ec:cd:69:a8:33:4b:95:11:08:ea:d2
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.2.2 Dummy Root CA Certificate – hash

The SHA256 hash of the DER-encoded dummy Root CA Certificate data (shown above) is as follows:

```
0xcb290519c6526794c24dd53bcd15c20f3996a8ac62b28f7591444677b39c0a9c
```

This value, the 'Root digest', is used to identify the Root CA Certificate in Product Unit Certificate Chains.

8.2.3 Dummy Root CA Certificate – data

Parsing the DER-encoded, ASN.1 certificate data above yields an X.509 certificate containing the following tbsCertificate field values and signatureValue data.

```
            serialNumber: 0x2f1b24bdeeb032bd (8 bytes)
                  issuer: 'WPCCAX' (commonName)
       validity.notBefore: 2000-Jan-01 00:00:00 (UTCTime, ignored)
        validity.notAfter: 9999-Dec-31 23:59:59 (GeneralizedTime, ignored)
                 subject: 'WPCCAX' (commonName)
       subjectPublicKey: 04:52:aa:a0:fb:0c:4f:08:91:58:1e:0f:c6:b6:f9:a9:
                         3d:ae:ea:8b:95:32:eb:9d:c9:c5:ef:98:5d:fd:80:d3:
                         1c:8d:8e:17:7e:af:e3:da:53:8a:db:56:d9:5d:66:90:
                         8f:2b:43:7c:59:3c:4c:34:ed:b7:c5:a1:ed:98:11:08:
                         f3 (65 bytes)
       basicConstraints: cA:True pathLen:None
              signature: 30:45:02:20:4c:79:ab:6e:34:04:0b:09:4d:a3:8b:1d:
                         6d:f4:02:73:c6:e5:b9:c8:08:fe:3a:b0:ff:22:cc:e2:
                         d9:b8:ab:c4:02:21:00:a6:96:00:09:2f:55:39:6f:62:
                         5a:9f:6e:cf:44:35:a8:3e:4a:53:52:91:ec:cd:69:a8:
                         33:4b:95:11:08:ea:d2 (71 bytes)
```

A fuller, 'byte -by-byte' interpretation of the DER-encoded data is shown in Section A.2, *Dummy Root CA Certificate in ASN.1 parser output*.

8.2.4 Verification of the Root CA Certificate signature

The Root CA Certificate must be acquired using a secure channel as, being self-signed, it includes no inherent proof of authenticity. The signature can still be checked, however, as a test of the integrity of the certificate (i.e. to check that the data of the "to be signed" portion of the certificate is exactly the same data that the Root CA signed).

Using the procedure outlined in Section 8.1.1, *Verifying the ECDSA signature of an X.509 Certificate* we can prepare the data for verification of the signature. We start by extracting three pieces if information from the certificate: the "tbsCertificate" portion of the raw certificate, the claimed "issuer" and the "signature" data (which we deserialize):

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

tbsCertificate data

```
30:81:d3:a0:03:02:01:02:02:08:2f:1b:24:bd:ee:b0:  32:bd:30:0a:06:08:2a:86:48:ce:3d:04:03:02:30:11:
31:0f:30:0d:06:03:55:04:03:0c:06:57:50:43:43:41:  58:30:20:17:0d:30:30:30:30:31:30:31:30:30:30:30:30:
30:5a:18:0f:39:39:39:39:31:32:33:31:32:33:35:39:  35:39:5a:30:11:31:0f:30:0d:06:03:55:04:03:0c:06:
57:50:43:43:41:58:30:59:30:13:06:07:2a:86:48:ce:  3d:02:01:06:08:2a:86:48:ce:3d:03:01:07:03:42:00:
04:52:aa:a0:fb:0c:4f:08:91:58:1e:0f:c6:b6:f9:a9:  3d:ae:ea:8b:95:32:eb:9d:c9:c5:ef:98:5d:fd:80:d3:
1c:8d:8e:17:7e:af:e3:da:53:8a:db:56:d9:5d:66:90:  8f:2b:43:7c:59:3c:4c:34:ed:b7:c5:a1:ed:98:11:08:
f3:a3:13:30:11:30:0f:06:03:55:1d:13:01:01:ff:04:  05:30:03:01:01:ff
```

Issuer

```
issuer = WPCCAX
```

Signature

```
r = 0x4c79ab6e34040b094da38b1d6df40273c6e5b9c808fe3ab0ff22cce2d9b8abc4 # signature.r (to be verified)
s = 0xa69600092f55396f625a9f6ecf4435a83e4a535291eccd69a8334b951108ead2 # signature.s (to be verified)
```

We then need to establish the “public key” of the claimed issuer. In this ‘self-signed’ case, where the “subject” and “issuer” of the certificate are the same, the “public key” is contained in the certificate (serialized as the data of the “subjectPublicKey” field). To obtain the “public key” we therefore need to extract the data of the “subjectPublicKey” field and deserialize it. Doing so yields the following public key:

Public key

```
x = 0x52aaa0fb0c4f0891581e0fc6b6f9a93daeea8b9532eb9dc9c5ef985dfd80d31c # public_key.x
y = 0x8d8e177eafe3da538adb56d95d66908f2b437c593c4c34edb7c5a1ed981108f3 # public_key.y
```

The full collection of data values that we feed into an ECC signature verification routine are therefore as shown below (including z – the SHA256 hash of the “tbsCertificate” data).

signature verification details

```
z = 0xb6a87e0052955c783a7a6757142a29b75bd8889360cc96a130e1c7feae9b68e0 # message hash (digest)
x = 0x52aaa0fb0c4f0891581e0fc6b6f9a93daeea8b9532eb9dc9c5ef985dfd80d31c # public_key.x
y = 0x8d8e177eafe3da538adb56d95d66908f2b437c593c4c34edb7c5a1ed981108f3 # public_key.y
r = 0x4c79ab6e34040b094da38b1d6df40273c6e5b9c808fe3ab0ff22cce2d9b8abc4 # signature.r (to be verified)
s = 0xa69600092f55396f625a9f6ecf4435a83e4a535291eccd69a8334b951108ead2 # signature.s (to be verified)
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.3 Manufacturer CA Certificate Example

In these examples, we consider a fictitious device manufacturer, the ACME Charger Company, who is a member of the WPC and has been assigned the Power Transmitter Product manufacturer code (PTMC) 0xCACA.

In a real manufacturing scenario, the ACME Charger Company would set up a secure production line complete with a hardware security module (HSM) that it would use to protect the private key it uses to sign Product Unit Certificates. In this example, however, we reveal the private key in order to expose all the inputs to the signature calculations. Here is that private ECC key, along with the associated public key.

```
public key.x: 0xb3d1ba7d2e614ff78cd663fa5070bcefba523c9d3ab1c8f0c5da6cf503de43e2
public key.y: 0x9c603050ec4b253eae640bcc38180aecf835c94f0bd8029211b5bb7aa7fa856f
```

As part of the process of becoming a Manufacturer CA, a manufacturer needs to obtain a Manufacturer CA Certificate from the WPC that it can include in Product Unit Certificate chains. To obtain this certificate, the manufacturer submits a Certificate Signing Request (CSR) to the WPC Root CA containing two pieces of information: a string to be used as an identifier for the Manufacturer CA (the 'subject' of the certificate) and a copy of the public key of the public-private key pair that the Manufacturer CA will use to generate signatures.

The two values included in the CSR sent by our example manufacturer are as follows:

```
                    subject: 'CACA-X1' (commonName)
            subjectPublicKey: 04:b3:d1:ba:7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:
                              ef:ba:52:3c:9d:3a:b1:c8:f0:c5:da:6c:f5:03:de:43:
                              e2:9c:60:30:50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:
                              ec:f8:35:c9:4f:0b:d8:02:92:11:b5:bb:7a:a7:fa:85:
                              6f (65 bytes)
```

The format of the subject string is defined in Table 6. The ACME Charger company must start the string with a copy of its PTMC hex value and has chosen 'X1' as the two suffix characters (that allow the particular ACME Charger Company Manufacturer CA Certificate to be identified).

ECC public keys need to be serialized to be included as the value of the subjectPublicKey of the CSR (and in all the Qi Authentication X.509 Certificates). This serialization can either use the classic point serialization format or the compressed point format. The compressed format is normally preferred, as it required less storage and less transmission bandwidth and time. In this example, however, we have used the uncompressed format for the Manufacturer CA key so that we can later generate an example of the longest possible Certificate Chain.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.3.1 Example Manufacturer CA Certificate — DER-encoded

The raw bytes of the DER-encoded, ASN.1 Manufacturer CA Certificate returned by the WPC Root CA to ACME Charger Company (in response to the CSR) are shown here. See Annex A.3, *Manufacturer CA Certificate in PEM Format*, for a PEM encoded version.

```
30:82:01:49:30:81:ef:a0:03:02:01:02:02:0a:00:ef:  f1:1d:6a:02:fe:a1:8f:d2:30:0a:06:08:2a:86:48:ce:
3d:04:03:02:30:11:31:0f:30:0d:06:03:55:04:03:0c:  06:57:50:43:43:41:58:30:22:18:0f:32:30:30:30:30:
31:30:31:30:30:30:30:30:30:5a:18:0f:39:39:39:39:  31:32:33:31:32:33:35:39:35:39:5a:30:12:31:10:30:
0e:06:03:55:04:03:0c:07:43:41:43:41:2d:58:31:30:  59:30:13:06:07:2a:86:48:ce:3d:02:01:06:08:2a:86:
48:ce:3d:03:01:07:03:42:00:04:b3:d1:ba:7d:2e:61:  4f:f7:8c:d6:63:fa:50:70:bc:ef:ba:52:3c:9d:3a:b1:
c8:f0:c5:da:6c:f5:03:de:43:e2:9c:60:30:50:ec:4b:  25:3e:ae:64:0b:cc:38:18:0a:ec:f8:35:c9:4f:0b:d8:
02:92:11:b5:bb:7a:a7:fa:85:6f:a3:2a:30:28:30:12:  06:03:55:1d:13:01:01:ff:04:08:30:06:01:01:ff:02:
01:00:30:12:06:05:67:81:14:01:01:01:01:ff:04:06:  04:04:00:00:00:01:30:0a:06:08:2a:86:48:ce:3d:04:
03:02:03:49:00:30:46:02:21:00:90:6e:b5:64:05:aa:  b1:b1:2e:af:f0:da:92:70:65:e6:df:5d:37:06:84:45:
95:71:70:71:97:58:e9:a9:6c:da:02:21:00:f8:8a:2c:  c4:d8:c1:b1:b6:9a:60:2d:f5:16:8a:7a:34:e4:8f:31:
3a:2a:48:4c:e6:63:62:07:1a:60:e8:95:29
```

8.3.2 Example Manufacturer CA Certificate - data

Parsing this DER-encoded, ASN.1 data yields an X.509 certificate containing the following tbsCertificate field values and signatureValue data.

```
               serialNumber: 0xeff11d6a02fea18fd2
                     issuer: 'WPCCAX' (commonName)
           validity.notBefore: 2000-Jan-01 00:00:00 (GeneralizedTime, ignored)
            validity.notAfter: 9999-Dec-31 23:59:59 (GeneralizedTime, ignored)
                    subject: 'CACA-X1' (commonName)
             subjectPublicKey: 04:b3:d1:ba:7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:
                               ef:ba:52:3c:9d:3a:b1:c8:f0:c5:da:6c:f5:03:de:43:
                               e2:9c:60:30:50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:
                               ec:f8:35:c9:4f:0b:d8:02:92:11:b5:bb:7a:a7:fa:85:
                               6f (65 bytes)
           basicConstraints: cA:True pathLen:0
               wpc-qi-policy: 00:00:00:01
                   signature: 30:46:02:21:00:90:6e:b5:64:05:aa:b1:b1:2e:af:f0:
                              da:92:70:65:e6:df:5d:37:06:84:45:95:71:70:71:97:
                              58:e9:a9:6c:da:02:21:00:f8:8a:2c:c4:d8:c1:b1:b6:
                              9a:60:2d:f5:16:8a:7a:34:e4:8f:31:3a:2a:48:4c:e6:
                              63:62:07:1a:60:e8:95:29 (72 bytes)
```

A fuller, 'byte -by-byte' interpretation of the DER-encoded data is shown in Annex A.4, *Manufacturer CA Certificate in ASN.1 parser output*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.3.3 Verification of the Manufacturer CA Certificate signature

Assuming the manufacturer received the Manufacturer CA Certificate through a secure channel, it has no reason to suspect the authenticity of this certificate. However, the manufacturer can check the integrity of the certificate by verifying its signature. A Power Receiver Product makes the same check for both integrity and authenticity. (Details of the verification of this signature are given in later in Section 8.6.1.5 , *Processing of the CERTIFICATE response data*.)

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.4 Example Product Unit Certificates

A manufacturer must provision each individual product unit with a unique Product Unit Certificate. Here we generate two sample Product Unit Certificates. The first uses all possible fields and the largest legal values for these fields, yielding a Product Unit Certificate of the maximum length possible. The second example represents a more typical length containing a subset of fields and non-maximum value lengths.

8.4.1 Example 1: maximum length Product Unit Certificate

In this first example, the values in the data fields will yield a Product Unit Certificate of maximum length. The Manufacturer CA Certificate generated earlier is also of the maximum length. When combining these with the Root CA Certificate hash to create a Certificate Chain (see Section 8.5, *Certificate Chain and digest of certificates example*), the Certificate Chain will also have the maximum length possible.

In a compliant product, the private key that an individual product uses for Authentication is a secret that would remain secure in its Secure Storage Subsystem. In these examples, however, we reveal the private keys of the product units (just as we did with the Manufacturer CA) to expose all the inputs to the signature calculations. Here is that private key ECC key for our first example product unit, along with the associated public key.

ECC private-public keys

```
x = 0x77b1f30e5d79a63fbcc35de8436e45d89c15f9998e8b8f2c6001caedae5f859  # public_key.x
y = 0x3a5076d2c7a4af0bc56b479de16ada110c0aefd739e1f04d0dd7657eb9321353 # public_key.y
```

Like the public key for the Manufacturer CA Certificate, the public key for this first Product Unit Certificate is encoded with the uncompressed format to create a sample Product Unit Certificate Chain of maximum length.

The following bytes are the serialized public key using the uncompressed point format

Encoded public key

```
04:07:7b:1f:30:e5:d7:9a:63:fb:cc:35:de:84:36:e4:  5d:89:c1:5f:99:98:e8:b8:f2:c6:00:1c:ae:da:e5:f8:
59:3a:50:76:d2:c7:a4:af:0b:c5:6b:47:9d:e1:6a:da:  11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:65:7e:b9:32:13:
53
```

In addition to the public key, the Product Unit Certificate contains several other data values (as defined in Table 7). Many of these (such as validity dates, serial numbers and RSIDs) would be automatically generated by the HSM or the software that uses it, other values (such as the Qi ID) would be static data for all units produced of the same type. The particular values used for these other fields in this example are unimportant – except that they were chosen to generate the maximum possible certificate length. These particular values can be seen later in the section *Parser output* on page 55.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Before we can assemble the full Product Unit Certificate, we need to bring the bytes of the encoded public key together with all these other data fields of the tbsCertificate portion of the Product Unit Certificate and DER-encode their ASN.1 representation so that we can generate the signature that will be included in the certificate.

tbsCertificate data

```
30:82:01:5b:a0:03:02:01:02:02:0a:00:b0:9b:9f:24:  00:a7:9e:a0:ae:30:0a:06:08:2a:86:48:ce:3d:04:03:
02:30:12:31:10:30:0e:06:03:55:04:03:0c:07:43:41:  43:41:2d:58:31:30:22:18:0f:32:30:32:31:30:38:32:
33:31:35:35:31:35:33:5a:18:0f:32:30:32:31:30:38:  32:34:31:35:35:31:35:33:5a:30:81:8b:31:2c:30:2a:
06:03:55:04:03:0c:23:30:30:30:31:32:33:2d:52:61:  70:69:64:20:63:68:61:72:67:69:6e:67:20:62:61:67:
65:6c:20:74:6f:61:73:74:65:72:31:29:30:27:06:03:  55:04:5c:04:20:53:58:4d:67:64:47:68:70:63:79:42:
68:62:69:42:46:59:58:4e:30:5a:58:49:67:52:57:64:  6e:50:77:3d:3d:31:30:30:2e:06:0a:09:92:26:89:93:
f2:2c:64:01:01:0c:20:44:6f:20:6e:6f:74:20:75:73:  65:20:61:73:20:61:20:66:6c:6f:74:61:74:69:6f:6e:
20:64:65:76:69:63:65:30:59:30:13:06:07:2a:86:48:  ce:3d:02:01:06:08:2a:86:48:ce:3d:03:01:07:03:42:
00:04:07:7b:1f:30:e5:d7:9a:63:fb:cc:35:de:84:36:  e4:5d:89:c1:5f:99:98:e8:b8:f2:06:00:1c:ae:da:e5:
f8:59:3a:50:76:d2:c7:a4:af:0b:c5:6b:47:9d:e1:6a:  da:11:0c:0a:ef:d7:39:e1:f0:dd:0d:d7:65:7e:b9:32:
13:53:a3:1b:30:19:30:17:06:05:67:81:14:01:02:01:  01:ff:04:0b:04:09:f1:02:d3:c4:15:06:e7:68:79
```

We can now generate the signature that will be included in the Product Unit Certificate by generating an ECDSA signature of the tbsCertificate data with the private key associated with the Manufacturer CA Certificate. The resulting signature (r,s) values are shown here, together with all the values required to re-generate them, including secure random number (k) and the SHA256 hash of the tbsCertificate data (z):

signature details

```
z = 0x74eb9e9389f3ae6343443660869ff0e1e57373170dc16800c9b39e051b77cb1a # message (digest)
r = 0xa8290ec3c9afdc085258cbb47ab7023abcd5cd3678f41dce2f7c4ccc9546fc56 # signature.r
s = 0xfaeb67f91479de6d31deec0bb0c513e0a361dc449f7faf4e933fe904957adec10 # signature.s
```

The message digest (z) is the SHA256 hash of the DER-encoded, tbsCertificate data.

The final step in creating the DER-encoded, ASN.1 sequence that represents the full X.509 v3 certificate requires assembling the tbsCertificate data along with encoded metadata describing the signature scheme and an encoded form of the signature itself according to the X.509 v3 schema. Doing so leads to the fully assembled, DER-encoded ASN.1 Product Unit Certificate shown here:

Product Unit Certificate — example 1

To generate the full Product Unit Certificate, bring the bytes of the public key together with all the other data fields of the tbsCertificate portion of the Product Unit Certificate and assemble the generated DER-encoded, ASN1 representation.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

```
30:82:01:b6:30:82:01:5b:a0:03:02:01:02:02:0a:00:  b0:9b:9f:24:00:a7:9e:a0:ae:30:0a:06:08:2a:86:48:
ce:3d:04:03:02:30:12:31:10:30:0e:06:03:55:04:03:  0c:07:43:41:43:41:2d:58:31:30:22:18:0f:32:30:32:
31:30:38:32:33:31:35:35:31:35:33:5a:18:0f:32:30:  32:31:30:38:32:34:31:35:35:31:35:33:5a:30:81:8b:
31:2c:30:2a:06:03:55:04:03:0c:23:30:30:30:31:32:  33:2d:52:61:70:69:64:20:63:68:61:72:67:69:6e:67:
20:62:61:67:65:6c:20:74:6f:61:73:74:65:72:31:29:  30:27:06:03:55:04:5c:04:20:53:58:4d:67:64:47:68:
70:63:79:42:68:62:69:42:46:59:58:4e:30:5a:58:49:  67:52:57:64:6e:50:77:3d:3d:31:30:30:2e:06:0a:09:
92:26:89:93:f2:2c:64:01:01:0c:20:44:6f:20:6e:6f:  74:20:75:73:65:20:61:73:20:61:20:66:6c:6f:74:61:
74:69:6f:6e:20:64:65:76:69:63:65:30:59:30:13:06:  07:2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:
01:07:03:42:00:04:07:7b:1f:30:e5:d7:9a:63:fb:cc:  35:de:84:36:e4:5d:89:c1:5f:99:98:e8:b8:f2:c6:00:
1c:ae:da:e5:f8:59:3a:50:76:d2:c7:a4:af:0b:c5:6b:  47:9d:e1:6a:da:11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:
65:7e:b9:32:13:53:a3:1b:30:19:30:17:06:05:67:81:  14:01:02:01:01:ff:04:0b:04:09:f1:02:d3:c4:15:06:
e7:68:79:30:0a:06:08:2a:86:48:ce:3d:04:03:02:03:  49:00:30:46:02:21:00:a8:29:0e:c3:c9:af:dc:08:52:
58:cb:b4:7a:b7:02:3a:bc:d5:cd:36:78:f4:1d:ce:2f:  7c:4c:cc:95:46:fc:56:02:21:00:fa:eb:67:f9:14:79:
de:6d:31:de:e7:0b:0c:51:3e:0a:36:1d:c4:49:f7:fa:  f4:e9:33:fe:90:49:57:ad:ec:10
```

A PEM-encoded version of this certificate is given in Annex A.5, *Product Unit Certificate, example 1 in PEM format*.

As we used the maximum sizes for all certificate fields values (and values that lead to the maximum encoded lengths), the length of this certificate example (442 bytes) is the maximum possible length for a Product Unit Certificate.

Parser output

Parsing this DER-encoded, ASN.1 data yields an X.509 certificate containing the following tbsCertificate field values and signatureValue data.

```
          serialNumber: 0xb09b9f2400a79ea0ae (9 bytes, 10 when encoded)
                issuer: 'CACA-X1' (commonName)
    validity.notBefore: 2021-Aug-23 15:51:53 (GeneralizedTime, ignored)
     validity.notAfter: 2021-Aug-24 15:51:53 (GeneralizedTime, ignored)
    subject.attribute1: '000123-Rapid charging bagel toaster' (commonName, 35 chars [6+1+28])
                        qi_id: 123
                        model: 'Rapid charging bagel toaster' (28 chars)
    subject.attribute2: 53:58:4d:67:64:47:68:70:63:79:42:68:62:69:42:46:
                        59:58:4e:30:5a:58:49:67:52:57:64:6e:50:77:3d:3d (tagAFI, 32 bytes)
    subject.attribute3: 'Do not use as a flotation device' (userId, 32 chars)
      subjectPublicKey: 04:07:7b:1f:30:e5:d7:9a:63:fb:cc:35:de:84:36:e4:
                        5d:89:c1:5f:99:98:e8:b8:f2:c6:00:1c:ae:da:e5:f8:
                        59:3a:50:76:d2:c7:a4:af:0b:c5:6b:47:9d:e1:6a:da:
                        11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:65:7e:b9:32:13:
                        53 (65 bytes)
           wpc-qi-rsid: f1:02:d3:c4:15:06:e7:68:79 (9 bytes)
             signature: 30:46:02:21:00:a8:29:0e:c3:c9:af:dc:08:52:58:cb:
                        b4:7a:b7:02:3a:bc:d5:cd:36:78:f4:1d:ce:2f:7c:4c:
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

```
                    cc:95:46:fc:56:02:21:00:fa:eb:67:f9:14:79:de:6d:
                    31:de:e7:0b:0c:51:3e:0a:36:1d:c4:49:f7:fa:f4:e9:
                    33:fe:90:49:57:ad:ec:10 (72 bytes)
```

A fuller, 'byte -by-byte' interpretation of the DER-encoded data is shown in Annex A.6, *Product Unit Certificate, example 1 in ASN.1 parser output*.

8.4.2 Example 2: typical length Product Unit Certificate

In this second example, the Product Unit Certificate omits some of the optional fields, uses less than the maximum length for variable length fields, and uses the compressed point format for its public key. This example is likely to be more typical of certificates encountered in real products than the first example.

Here is that private key ECC key that we will use for the second example product unit, along with the associated public key.

ECC private-public keys

```
x = 0xefc57d5561496d90551e2f74c829520c360689d0f05bdef5f3d6e64a4639d71f # public_key.x
y = 0xd6cbeaa0152853f7e2981ccff77928ba8055e416ad04f9725e7eba724d5a7c14 # public_key.y
```

For details of the values used in the certificate other than the public key (shown above), see *parsed output*. The DER-encoded, ASN.1 formatted tbsCertificate portion of the second certificate is shown here.

tbsCertificate data

```
30:81:c5:a0:03:02:01:02:02:09:00:f9:d5:c4:77:48:  c4:eb:11:30:0a:06:08:2a:86:48:ce:3d:04:03:02:30:
12:31:10:30:0e:06:03:55:04:03:0c:07:48:41:43:41:  2d:58:31:30:22:18:0f:32:30:32:31:30:38:32:34:31:
32:30:36:30:30:5a:18:0f:32:30:32:31:30:38:32:35:  31:32:30:36:30:30:5a:30:18:31:16:30:14:06:03:55:
04:03:0c:0d:30:30:30:30:34:32:2d:4d:6f:64:65:6c:  33:30:39:30:13:06:07:2a:86:48:ce:3d:02:01:06:08:
2a:86:48:ce:3d:03:01:07:03:22:00:02:ef:c5:7d:55:  61:49:6d:90:55:1e:2f:74:c8:29:52:0c:36:06:89:d0:
f0:5b:de:f5:f3:d6:e6:4a:a6:39:d7:1f:a3:1a:30:18:  30:16:06:05:67:81:14:01:02:01:01:ff:04:0a:04:08:
00:00:de:ad:be:ef:12:34
```

As with the first example, we generate the signature that will be included in the Product Unit Certificate by generating an ECDSA signature of the tbsCertificate data with the private key associated with the Manufacturer CA Certificate. The resulting signature (***r,s***) values are shown here, together will all the values required to re-generate them, including secure random number (***k***) and the SHA256 hash of the tbsCertificate data (***z***):

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

signature details

```
z = 0xc348c7cb55b4039c96fd033685ef4aeaf56b648b2011d48fb164685806aa4f73 # message (digest)
r = 0x6650b1e260d662a10b757051146dbe110fcd5cf92b1fff340b18896fea958569 # signature.r
s = 0xaac30f10cb8a4a0055a3ccec0c94d5e031ad45a919b5b28f29a63e58c7097b0f # signature.s
```

To create the DER-encoded, ASN.1 sequence to yield the following full X.509 v3 certificate, assemble the tbsCertificate data along with encoded metadata describing the signature scheme and an encoded form of the signature itself.

Assembling the tbsCertificate data along with encoded metadata describing the signature scheme and an encoded form of the signature data according to the X.509 v3 schema yields the full, DER-encoded, ASN.1 example 2 Product Unit Certificate shown here

Product Unit Certificate — example 2

```
30:82:01:1e:30:81:c5:a0:03:02:01:02:02:09:00:f9:  d5:c4:77:48:c4:eb:11:30:0a:06:08:2a:86:48:ce:3d:
04:03:02:30:12:31:10:30:0e:06:03:55:04:03:0c:07:  43:41:43:41:2d:58:31:30:32:18:0f:32:30:32:31:30:
38:32:34:31:32:30:36:30:30:5a:18:0f:32:30:32:31:  30:38:32:35:31:32:30:36:30:30:5a:30:18:31:16:30:
14:06:03:55:04:03:0c:0d:30:30:30:30:34:32:2d:4d:  6f:64:65:6c:33:30:39:30:13:06:07:2a:86:48:ce:3d:
02:01:06:08:2a:86:48:ce:3d:03:01:07:03:22:00:02:  ef:c5:7d:55:61:49:6d:90:55:1e:2f:74:c8:29:52:0c:
36:06:89:d0:f0:5b:de:f5:f3:d6:e6:4a:a6:39:d7:1f:  a3:1a:30:18:30:16:06:05:67:81:14:01:02:01:01:ff:
04:0a:04:08:00:00:de:ad:be:ef:12:34:30:0a:06:08:  2a:86:48:ce:3d:04:03:02:03:48:00:30:45:02:20:66:
50:b1:e2:60:d6:62:a1:0b:75:70:51:14:6d:be:11:0f:  cd:3c:f9:2b:1f:ff:34:0b:18:89:6f:ea:95:85:69:02:
21:00:aa:c3:0f:10:cb:8a:4a:00:55:a3:cc:ec:0c:94:  d5:e0:31:ad:45:a9:19:b5:b2:8f:29:a6:3e:58:c7:09:
7b:0f:
```

A PEM-encoded version of this certificate is given in Annex A.7, *Product Unit Certificate, example 2 in PEM format*.

Parsing this DER-encoded, ASN.1 data yields an X.509 certificate containing the following tbsCertificate field values and signatureValue data.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

parsed output

```
                 serialNumber: 0xf9d5c47748c4eb11 (8 bytes, 9 when encoded)
                      issuer: 'CACA-X1' (commonName)
           validity.notBefore: 2021-Aug-24 12:06:00 (GeneralizedTime, ignored)
            validity.notAfter: 2021-Aug-25 12:06:00 (GeneralizedTime, ignored)
           subject.attribute1: '000042-Model3' (commonName, 13 chars [6+1+6])
                               qi_id: 42
                               model: 'Model3' (6 chars)
           subject.attribute2: - not used -
           subject.attribute3: - not used -
             subjectPublicKey: 02:ef:c5:7d:55:61:49:6d:90:55:1e:2f:74:c8:29:52:
                               0c:36:06:89:d0:f0:5b:de:f5:f3:d6:e6:4a:a6:39:d7:
                               1f (33 bytes)
                 wpc-qi-rsid: 00:00:de:ad:be:ef:12:34 (8 bytes)
                   signature: 30:45:02:20:66:50:b1:e2:60:d6:62:a1:0b:75:70:51:
                               14:6d:be:11:0f:cd:5c:f9:2b:1f:ff:34:0b:18:80:6f:
                               ea:95:85:69:02:21:00:aa:c3:0f:10:cb:8a:4a:00:55:
                               a3:cc:ec:0c:94:d5:e0:31:ad:45:a9:19:57:b2:8f:29:
                               a6:3e:58:c7:09:7b:0f (71 bytes)
```

A fuller, 'byte -by-byte' interpretation of the DER-encoded data is shown in Annex A.8, *Product Unit Certificate, example 2 in ASN.1 parser output*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.5 Certificate Chain and digest of certificates example

As defined in Table 7 (in Section 3.1), assembling a Certificate Chain is the process of:

- Concatenating the bytes of the Root CA Certificate hash together with the bytes of the DER-encoded, Manufacturer CA Certificate, and those of the DER-encoded Product Unit Certificate
- Prepending that whole sequence with two bytes that indicate the total sequence length (including the two bytes of the length code itself)

8.5.1 Example 1: maximum length Certificate Chain

In this first example of generating a Certificate Chain, we assemble the chain for the first example Product Unit Certificate we created.

- Concatenating the 32 bytes of the Root CA Certificate hash (see Section 8.2.2, *Dummy Root CA Certificate – hash*), the 333 bytes of our ACME Charger Co example Manufacturer CA Certificate (see Section 8.3.1, *Example Manufacturer CA Certificate — DER-encoded*) and the 442 bytes of the first example Product Unit Certificate (see Chapter 8, *Example Product Unit Certificates*) yields a sequence of 807 bytes (32 + 333 + 442).
- Prepending this with two bytes that indicate the final sequence length (809 = 2 + 807) yields the following complete Certificate Chain byte sequence.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

```
03:29:cb:29:05:19:c6:52:67:94:c2:4d:d5:3b:cd:15:c2:0f:39:96:a8:ac:62:b2:8f:75:91:44:46:77:b3:9c:
0a:9c:30:82:01:49:30:81:ef:a0:03:02:01:02:02:0a:00:ef:f1:1d:6a:02:fe:a1:8f:d2:30:0a:06:08:2a:86:
48:ce:3d:04:03:02:30:11:31:0f:30:0d:06:03:55:04:03:0c:06:57:50:43:43:41:58:30:22:18:0f:32:30:30:
30:30:31:30:31:30:30:30:30:30:30:5a:18:0f:39:39:39:39:31:32:33:31:32:33:35:39:35:39:5a:30:12:31:
10:30:0e:06:03:55:04:03:0c:07:43:41:43:41:2d:58:31:30:59:30:13:06:07:2a:86:48:ce:3d:02:01:06:08:
2a:86:48:ce:3d:03:01:07:03:42:00:04:b3:d1:ba:7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:ef:ba:52:3c:9d:
3a:b1:c8:f0:c5:da:6c:f5:03:de:43:e2:9c:60:30:50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:ec:f8:35:c9:4f:
0b:d8:02:92:11:b5:bb:7a:a7:fa:85:6f:a3:2a:30:28:30:12:06:03:55:1d:13:01:01:ff:04:08:30:06:01:01:
ff:02:01:00:30:12:06:05:67:81:14:01:01:01:01:ff:04:06:04:04:00:00:00:01:30:0a:06:08:2a:86:48:ce:
3d:04:03:02:03:49:00:30:46:02:21:00:90:6e:b5:64:05:aa:b1:b1:2e:af:f0:da:92:70:65:e6:df:5c:97:06:
84:45:95:71:70:71:97:58:e9:a9:6c:da:02:21:00:f8:8a:2c:c4:d8:c1:b1:b6:9a:60:2d:f5:16:8e:7a:34:e4:
8f:31:3a:2a:48:4c:e6:63:62:07:1a:60:e8:95:29:30:82:01:b6:30:82:01:5b:a0:03:02:01:02:02:0a:00:b0:
9b:9f:24:00:a7:9e:a0:ae:30:0a:06:08:2a:86:48:ce:3d:04:03:02:30:12:31:10:30:0e:06:03:55:04:03:0c:
07:43:41:43:41:2d:58:31:30:22:18:0f:32:30:32:31:30:38:32:33:31:35:35:31:35:33:5a:18:0f:32:30:32:
31:30:38:32:34:31:35:35:31:35:33:5a:30:81:8b:31:2c:30:2a:06:03:55:04:03:0c:23:30:30:30:31:32:33:
2d:52:61:70:69:64:20:63:68:61:72:67:69:6e:67:20:62:61:67:65:6c:20:74:6f:61:73:74:65:72:31:29:30:
27:06:03:55:04:5c:04:20:53:58:4d:67:64:47:68:70:63:79:42:68:62:69:42:46:59:58:4e:30:5a:58:49:67:
52:57:64:6e:50:77:3d:3d:31:30:30:2e:06:0a:09:92:26:89:93:f2:2c:64:01:01:0c:20:44:6f:20:6e:6f:74:
20:75:73:65:20:61:73:20:61:20:66:6c:6f:74:61:74:69:6f:6e:20:64:65:76:69:63:65:30:59:30:13:06:07:
2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:01:07:03:42:00:04:07:7b:1f:30:e5:d7:9a:63:fb:cc:35:
de:84:36:e4:5d:89:c1:5f:99:98:e8:b8:f2:c6:00:1c:ae:da:e5:f8:59:3a:50:76:d2:c7:a4:af:0b:c5:6b:47:
9d:e1:6a:da:11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:6d:7e:b9:32:13:53:a3:1b:30:19:30:17:06:05:67:81:14:
01:02:01:01:ff:04:0b:04:09:f1:02:d3:c4:15:06:e7:68:79:30:0a:06:08:2a:86:48:ce:3d:04:03:02:03:49:
00:30:46:02:21:00:a8:29:0e:c3:c9:af:dc:08:52:58:cb:b4:7a:b7:02:3a:bc:d5:cd:36:78:f4:1d:ce:2f:7c:
4c:cc:95:46:fc:56:02:21:00:fa:eb:67:f9:14:79:de:6d:31:de:e7:0b:0c:51:3e:0a:36:1d:c4:49:f7:fa:f4:
e9:33:fe:90:49:57:ad:ec:10
```

As both the Manufacturer CA Certificate and the Product Unit Certificate in this chain are of the maximum possible lengths (and the lengths of the other two components are fixed), this is an example of a Certificate Chain of the maximum possible size (809 bytes).

Certificate Chain digest

The SHA256 hash "digest" of each Certificate Chain is used as an identifier in various parts of the authentication protocol. The SHA256 hash of this sample chain is:

```
0x4629653ad1ceb37c6a36f0cc11b4291686392785f0f826dfded35eac5fcc50fc
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.5.2 Example 2: typical length Certificate Chain

Our second example of generating a Certificate Chain assembles the chain for the second example Product Unit Certificate we created. Other than which Product Unit Certificate is used (and therefore the final length of the chain), this example is the same as the first one. The second chain is shown here and has total length of 657 (2 + 32 + 333 + 290).

```
02:91:cb:29:05:19:c6:52:67:94:c2:4d:d5:3b:cd:15:c2:0f:39:96:a8:ac:62:b2:8f:75:91:44:46:77:b3:9c:
0a:9c:30:82:01:49:30:81:ef:a0:03:02:01:02:02:0a:00:ef:f1:1d:6a:02:fe:a1:8f:d2:30:0a:06:08:2a:86:
48:ce:3d:04:03:02:30:11:31:0f:30:0d:06:03:55:04:03:0c:06:57:50:43:43:41:58:30:22:18:0f:32:30:30:
30:30:31:30:31:30:30:30:30:30:30:5a:18:0f:39:39:39:39:31:32:33:31:32:33:35:39:35:39:5a:30:12:31:
10:30:0e:06:03:55:04:03:0c:07:43:41:43:41:2d:58:31:30:59:30:13:06:07:2a:86:48:ce:3d:02:01:06:08:
2a:86:48:ce:3d:03:01:07:03:42:00:04:b3:d1:ba:7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:ef:ba:52:3c:9d:
3a:b1:c8:f0:c5:da:6c:f5:03:de:43:e2:9c:60:30:50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:ec:f8:35:c9:4f:
0b:d8:02:92:11:b5:bb:7a:a7:fa:85:6f:a3:2a:30:28:30:12:06:03:55:1d:13:01:01:ff:04:08:30:06:01:01:
ff:02:01:00:30:12:06:05:67:81:14:01:01:01:01:ff:04:06:04:04:00:00:00:01:30:0a:06:08:2a:86:48:ce:
3d:04:03:02:03:49:00:30:46:02:21:00:90:6e:b5:64:05:aa:b1:b1:2e:af:f0:da:92:70:65:e6:df:5d:37:06:
84:45:95:71:70:71:97:58:e9:a9:6c:da:02:21:00:f8:8a:2c:c4:d8:c1:b1:b6:9a:60:2d:f5:16:8a:7a:34:e4:
8f:31:3a:2a:48:4c:e6:63:62:07:1a:60:e8:95:29:30:82:01:1e:30:81:c5:a0:03:02:01:02:02:09:00:f9:d5:
c4:77:48:c4:eb:11:30:0a:06:08:2a:86:48:ce:3d:04:03:02:30:12:31:10:30:0e:06:03:55:04:03:0c:07:43:
41:43:41:2d:58:31:30:22:18:0f:32:30:32:31:30:38:32:34:31:32:30:36:30:30:5a:18:0f:32:30:32:31:30:
38:32:35:31:32:30:36:30:30:5a:30:18:31:16:30:14:06:03:55:04:03:0c:0d:30:30:30:30:30:34:32:2d:4d:6f:
64:65:6c:33:30:39:30:13:06:07:2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:01:07:03:22:00:02:ef:
c5:7d:55:61:49:6d:90:55:1e:2f:74:c8:29:52:0c:36:0d:89:d0:f0:5b:de:f5:f3:d6:e6:4a:a6:39:d7:1f:a3:
1a:30:18:30:16:06:05:67:81:14:01:02:01:01:ff:04:0a:04:08:00:00:de:ad:be:ef:12:34:30:0a:06:08:2a:
86:48:ce:3d:04:03:02:03:48:00:30:45:02:20:66:50:b1:e2:60:d6:62:a1:0b:75:70:51:14:6d:be:11:0f:cd:
5c:f9:2b:1f:ff:34:0b:18:89:6f:ea:95:85:69:02:21:00:aa:c3:0f:10:cb:8a:4a:00:55:a3:cc:ec:0c:94:d5:
    e0:31:ad:45:a9:19:b5:b2:8f:28:a6:3e:58:c7:09:7b:0f
```

Certificate Chain digest

The SHA256 hash of this example chain is:

```
0xe36b91faf190e0874ce8656c28e23376e8ba29bdfbcfaf3cb34a81dd48847f22
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.6 Authentication examples

Here we demonstrate the details of the authentication of Power Transmitter Products by Power Receiver Products using the Authentication Protocol. The Power Transmitter Products in these examples use the example Certificate Chains that we generated earlier in the chapter.

8.6.1 Test of example 1, maximum length Certificate Chain

In this first authentication exchange example, a Power Receiver Product is going to use the "Flow with caching" behavior (described in Section 7.2, *Flow with caching*) to test the authenticity of a Power Transmitter Product that has the long Certificate Chain (described in Section 8.5.1, *Example 1: maximum length Certificate Chain*) loaded in Certificate Chain slot 0.

8.6.1.1 GET_DIGESTS request

Once the Power Receiver Product has confirmed that the Power Transmitter Product supports authentication, the exchange of messages begins with a GET_DIGESTS request from the Power Receiver Product. The Power Receiver Product chooses to ask for the digests of all slots and queries Power Transmitter Product for chain digests by sending a GET_DIGESTS request message composed as defined in Table 26.

Table 26: GET_DIGESTS request

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x19 (GET_DIGESTS Message Header)							
B_1	0x0 (Reserved)				0xF (Slot Mask)			

The concatenated hex form of this request is:

```
19:0f
```

8.6.1.2 DIGESTS response

The Power Transmitter Product returns a DIGESTS response message composed according to the definition given in Table 17 (in Section 5.1). As the Power Transmitter Product only has a single Certificate Chain (populated in slot 0) it sets both slot masks of the message to 0x1 and includes the 32-byte SHA256 digest of its slot 0 Certificate Chain as the 'digests returned' data (i.e. the digest value given in Section 8.5.1, *Example 1: maximum length Certificate Chain*.

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 27: DIGESTS response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x11 (DIGESTS Message Header)							
B_1	0x1 (Slots Populated Mask)				0x1 (Slots Returned Mask)			
B_2 … B_{33}	0x4629653ad1ceb37c6a36f0cc11b4291686392785f0f826dfded35eac5fcc50fc (Digests Returned)							

The concatenated hex form of this DIGESTS response message is:

```
11:11:46:29:65:3a:d1:ce:b3:7c:6a:36:f0:cc:11:b4:  29:16:86:39:27:85:f0:f8:26:df:de:d3:5e:ac:5f:cc:
50:fc
```

8.6.1.3 GET_CERTIFICATE request

After discovering that it does not have cached Certificate Chain data associated with the digest value returned in the DIGESTS response, the Power Receiver Product then requests the whole of the Certificate Chain in slot 0 of the Power Transmitter Product by sending a GET_CERTIFICATE request message shown in Table 28 (see this also described in Table 14 (in Section 5.1).

Table 28: GET_CERTIFICATE request message - example 1

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x1a (GET_CERTIFICATE Message Header)							
B_1 to B_{32}	0x0 (OffsetA8)			0x0 (LengthA8)			0x0 (Slot Number)	
B_{33} to B_{50}	0x00 (Offset70)							
B_{51} to B_{53}	0x00 (Length70)							

The concatenated hex form of this GET_CERTIFICATE request message is:

```
1a:00:00:00
```

8.6.1.4 CERTIFICATE response

On receipt of this GET_CERTIFICATE request, the Power Transmitter responds with a CERTIFICATE response using the format defined in Table 18 (in Section 5.3). As the request is for the full slot 0 Certificate Chain, the message is simply composed of the CERTIFICATE Message Header byte (0x12) followed by the bytes of the entire Certificate Chain held in slot 0. The concatenated hex form of this response is:

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

```
12:03:29:cb:29:05:19:c6:52:67:94:c2:4d:d5:3b:cd:  15:c2:0f:39:96:a8:ac:62:b2:8f:75:91:44:46:77:b3:
9c:0a:9c:30:82:01:49:30:81:ef:a0:03:02:01:02:02:  0a:00:ef:f1:1d:6a:02:fe:a1:8f:d2:30:0a:06:08:2a:
86:48:ce:3d:04:03:02:30:11:31:0f:30:0d:06:03:55:  04:03:0c:06:57:50:43:43:41:58:30:22:18:0f:32:30:
30:30:30:31:30:31:30:30:30:30:30:30:5a:18:0f:39:  39:39:39:31:32:33:31:32:33:35:39:35:39:5a:30:12:
31:10:30:0e:06:03:55:04:03:0c:07:43:41:43:41:2d:  58:31:30:59:30:13:06:07:2a:86:48:ce:3d:02:01:06:
08:2a:86:48:ce:3d:03:01:07:03:42:00:04:b3:d1:ba:  7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:ef:ba:52:3c:
9d:3a:b1:c8:f0:c5:da:6c:f5:03:de:43:e2:9c:60:30:  50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:ec:f8:35:c9:
4f:0b:d8:02:92:11:b5:bb:7a:a7:fa:85:6f:a3:2a:30:  28:30:12:06:03:55:1d:13:01:01:ff:04:08:30:06:01:
01:ff:02:01:00:30:12:06:05:67:81:14:01:01:01:01:  ff:04:06:04:04:00:00:00:01:30:0a:06:08:2a:86:48:
ce:3d:04:03:02:03:49:00:30:46:02:21:00:90:6e:b5:  64:05:aa:b1:b1:2e:af:f0:da:92:70:65:e6:df:5c:37:
06:84:45:95:71:70:71:97:58:e9:a9:6c:da:02:21:00:  f8:8a:2c:c4:d8:c1:b1:b6:9a:60:2d:f5:16:8a:7a:34:
e4:8f:31:3a:2a:48:4c:e6:63:62:07:1a:60:e8:95:29:  30:82:01:b6:30:82:01:5b:a0:03:02:01:02:02:0a:00:
b0:9b:9f:24:00:a7:9e:a0:ae:30:0a:06:08:2a:86:48:  ce:3d:04:03:02:30:12:31:10:30:0e:06:03:55:04:03:
0c:07:43:41:43:41:2d:58:31:30:22:18:0f:32:30:32:  31:30:38:32:33:31:35:35:31:35:33:5a:18:0f:32:30:
32:31:30:38:32:34:31:35:35:31:35:33:5a:30:81:8b:  31:2c:30:2a:06:03:55:04:03:0c:23:30:30:30:31:32:
33:2d:52:61:70:69:64:20:63:68:61:72:67:69:6e:67:  20:62:61:67:65:6c:20:74:6f:61:73:74:65:72:31:29:
30:27:06:03:55:04:5c:04:20:53:58:4d:67:64:47:68:  70:63:79:42:68:62:69:42:46:59:58:4e:30:5a:58:49:
67:52:57:64:6e:50:77:3d:3d:31:30:30:2e:06:0a:09:  92:26:89:93:f2:2c:64:01:01:0c:20:44:6f:20:6e:6f:
74:20:75:73:65:20:61:73:20:61:20:66:6c:6f:74:61:  74:69:6f:6e:20:64:65:76:69:63:65:30:59:30:13:06:
07:2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:  01:07:03:42:00:04:07:7b:1f:30:e5:d7:9a:63:fb:cc:
35:de:84:36:e4:5d:89:c1:5f:99:98:e8:b8:f2:c6:00:  1c:ae:da:e5:f8:59:3a:50:76:d2:c7:a4:af:0b:c5:6b:
47:9d:e1:6a:da:11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:  60:7e:b9:32:13:53:a3:1b:30:19:30:17:06:05:67:81:
14:01:02:01:01:ff:04:0b:04:09:f1:02:d3:c4:15:06:  e7:68:79:30:0a:06:08:2a:86:48:ce:3d:04:03:02:03:
49:00:30:46:02:21:00:a8:29:0e:c3:c9:af:dc:08:52:  58:cb:b4:7a:b7:02:3a:bc:d5:cd:36:78:f4:1d:ce:2f:
7c:4c:cc:95:46:fc:56:02:21:00:fa:eb:67:f9:14:79:  de:6d:31:de:e7:0b:0c:51:3e:0a:36:1d:c4:49:f7:fa:
f4:e9:33:fe:90:49:57:ad:ec:10
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.6.1.5 Processing of the CERTIFICATE response data

On receiving the Certificate Chain, the Power Receiver Product needs to split the chain into the Root CA Certificate digest, the Manufacturer CA Certificate, and the Product Unit Certificate. It then starts the process of verifying the authenticity of the Product Unit Certificate by checking that the Root CA Certificate Digest is the digest of a Root CA Certificate that it implicitly trusts. If it is, the Power Receiver Product then verifies that Manufacturer CA Certificate was signed by the private key associated with that Root CA Certificate, and finally that the Product Unit Certificate was signed by the private key associated with the Manufacturer CA Certificate.

Verification of the Root CA Certificate

The Power Receiver Product implicitly trusts one or more WPC Root CA Certificates. The process of checking the Root CA Certificate portion of the Certificate Chain is simply to test if the Root CA Certificate hash included in the chain is the hash of one of the WPC Root CA Certificates that the Power Receiver Product trusts. This is a simple numerical comparison.

Verification of the Manufacturer CA Certificate

To check the authenticity and integrity of the Manufacturer CA Certificate, the Power Receiver Product checks the validity of the ECDSA signature of the certificate following the process outlined earlier in Section 8.1.1, *Verifying the ECDSA signature of an X.509 Certificate*. We begin the process by extracting three pieces of information from the certificate: the "tbsCertificate" portion of the DER-encoded certificate, the claimed "issuer" and the "signature" data (which we deserialize):

tbsCertificate data

```
30:81:ef:a0:03:02:01:02:02:0a:00:ef:f1:1d:6a:02:  fe:a1:8f:d2:30:0a:06:08:2a:86:48:ce:3d:04:03:02:
30:11:31:0f:30:0d:06:03:55:04:03:0c:06:57:50:43:  43:41:58:30:22:18:0f:32:30:30:30:30:31:30:31:30:
30:30:30:30:30:5a:18:0f:39:39:39:39:31:32:33:31:  32:33:35:39:35:39:5a:30:12:31:10:30:0e:06:03:55:
04:03:0c:07:43:41:43:41:2d:58:31:30:59:30:13:06:  07:2a:86:48:ce:3d:02:01:06:08:2a:86:48:ce:3d:03:
01:07:03:42:00:04:b3:d1:ba:7d:2e:61:4f:f7:8c:d6:  63:fa:50:70:bc:ef:ba:52:3c:9d:3a:b1:c8:f0:c5:da:
6c:f5:03:de:43:e2:9c:60:30:50:ec:4b:25:3e:ae:64:  0b:cc:38:18:0a:ec:f8:35:c9:4f:0b:d8:02:92:11:b5:
bb:7a:a7:fa:85:6f:a3:2a:30:28:30:12:06:03:55:1d:  13:01:01:ff:04:08:30:06:01:01:ff:02:01:00:30:12:
06:05:67:81:14:01:01:01:01:ff:04:06:04:04:00:00:  00:01
```

issuer

```
WPCCAX
```

signature

```
r = 0x906eb56405aab1b12eaff0da927065e6df5d3706844595717071 9758e9a96cda # signature.r (to be verified)
s = 0xf88a2cc4d8c1b1b69a602df5168a7a34e48f313a2a484ce66362071a60e89529 # signature.s (to be verified)
```

We then need to establish the "public key" of the claimed issuer. Unlike the earlier, self-signed, Root CA example, the claimed "issuer" of the Manufacturer CA Certificate (the party that is mean to have signed it) is a second party (in this case "WPCCAX"). If the Power Receiver Product already trusts a CA called "WPCCAX" it will also know its "public key". In this case, "WPCCAX" represents the WPC Root CA – a party that the Power Receiver Product would implicitly trust and for which it would already know the "public key".

Starting the ECDSA verification process by generating the SHA256 hash of the "tbsCertificate" data then bringing this value (z) together with the "public key" of "WPCCAX" (as given in the Root CA Certificate and shown in serialized form in Section 8.2.3, *Dummy Root CA Certificate – data*) and the two components of the signature that is to be validated yields the full collection of data values (shown below) that we can then feed into an ECC signature verification routine.

signature verification details — Manufacturer CA Certificate

```
z = 0x8ad00ca5997f7effc86f315245ca3b460bbdb228721a63dfcf2a2b5a709f355b # message hash (digest)
x = 0x52aaa0fb0c4f0891581e0fc6b6f9a93daeea8b9532eb9dc9c5ef985dfd80d31c # public_key.x
y = 0x8d8e177eafe3da538adb56d95d66908f2b437c593c4c34edb7c5a1ed981108f3 # public_key.y
r = 0x906eb56405aab1b12eaff0da927065e6df5d3706844595717071 9758e9a96cda # signature.r (to be verified)
s = 0xf88a2cc4d8c1b1b69a602df5168a7a34e48f313a2a484ce66362071a60e89529 # signature.s (to be verified)
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

On verifying this signature, the Power Receiver Products assumes that it can trust the contents of the Manufacturer CA Certificate and caches the “public key” serialized in the certificate “subjectPublicKey” field as the trusted public key of Manufacturer CA “CACA-X1”.

Verification of the Product Unit Certificate

Now that the Power Receiver Product is satisfied with the integrity and authenticity of the Manufacturer CA Certificate, the Power Receiver Product then verifies the Product Unit Certificate following the same process. Once again, we extract and hash the tbsCertificate data and then, in this case, test the signature with respect to the public keys of CACA-X1—the claimed issuer of this certificate. The extracted values and the full collection of data input to the ECC signature verification algorithm are shown below.

tbsCertificate data

```
30:82:01:5b:a0:03:02:01:02:02:0a:00:b0:9b:9f:24:  00:a7:9e:a0:ae:30:0a:06:08:2a:86:48:ce:3d:04:03:
02:30:12:31:10:30:0e:06:03:55:04:03:0c:07:43:41:  43:41:2d:58:31:30:22:18:0f:32:30:32:31:30:38:32:
33:31:35:35:31:35:33:5a:18:0f:32:30:32:31:30:38:  32:34:31:35:35:31:35:33:5a:30:81:8b:31:2c:30:2a:
06:03:55:04:03:0c:23:30:30:30:31:32:33:2d:52:61:  70:69:64:20:63:68:61:72:67:69:6e:67:20:62:61:67:
65:6c:20:74:6f:61:73:74:65:72:31:29:30:27:06:03:  55:04:5c:04:20:53:58:4d:67:64:47:68:70:63:79:42:
68:62:69:42:46:59:58:4e:30:5a:58:49:67:52:57:64:  6e:50:77:3d:3d:31:30:30:2e:06:0a:09:92:26:89:93:
f2:2c:64:01:01:0c:20:44:6f:20:6e:6f:74:20:75:73:  65:20:61:73:20:61:20:66:6c:6f:74:61:74:69:6f:6e:
20:64:65:76:69:63:65:30:59:30:13:06:07:2a:86:48:  ce:3d:02:01:06:08:2a:86:48:ce:3d:03:01:07:03:42:
00:04:07:7b:1f:30:e5:d7:9a:63:fb:cc:35:de:84:36:  e4:5d:89:c1:ff:99:98:e8:b8:f2:c6:00:1c:ae:da:e5:
f8:59:3a:50:76:d2:c7:a4:af:0b:c5:6b:47:9d:e1:6a:  da:11:0c:0a:ef:d7:39:e1:f0:4d:0d:d7:65:7e:b9:32:
13:53:a3:1b:30:19:30:17:06:05:67:81:14:01:02:01:  01:ff:04:0b:04:09:f1:02:d3:c4:15:06:e7:68:79
```

Issuer

```
CACA-X1
```

signature verification details

```
z = 0x74eb9e9389f3ae6343443660869ff561e57373170dc16800c9b39e051b77cb1a # message hash (digest)
x = 0xb3d1ba7d2e614ff78cd663fa5070bcefba523c9d3ab1c8f0c5da6cf503de43e2 # public_key.x
y = 0x9c603050ec4b258eae640bcc38180aecf835c94f0bd8029211b5bb7aa7fa856f # public_key.y
r = 0xa8290ec39afdc085258cbb47ab7023abcd5cd3678f41dce2f7c4ccc9546fc56 # signature.r (to be verified)
s = 0xfaeb67f91479de6d31dee70b0c513e0a361dc449f7faf4e933fe904957adec10 # signature.s (to be verified)
```

On verifying this signature, the Power Receiver Product assumes that it can trust the contents of the Product Unit Certificate and caches the “public key” that is serialized in the certificate “subjectPublicKey” field as the trusted, “public key” of this individual product unit. (Securely caching this “public key” along with the SHA256 hash “digest” of the full Certificate Chain allows the parsing of this particular Certificate Chain to be skipped in the future.)

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.6.1.6 CHALLENGE request

When the Power Receiver Product is satisfied that the Certificate Chain is genuine, it continues the process of authentication by challenging the Power Transmitter Product to show that it has the private key associated with the Product Unit Certificate.

To do so, the Power Receiver Product sends a CHALLENGE request message asking for a signature to be generated using the private key associated with the slot that this Certificate Chain is held in (slot 0). The message, shown below in Table 29, includes some random data from the Power Receiver Product and is composed as defined in Table 15 (in Section 5.1).

Table 29: CHALLENGE request

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x1b (CHALLENGE Message Header)							
B_1	0x0 (Reserved)							0x0 (Slot Number)
B_2...B_{17}	0x3c2f44d265e340a5ccac0f724d792c4a (Nonce)							

The concatenated hex form of this is:

```
1b:00:3c:2f:44:d2:65:e3:40:a5:cc:ac:0f:72:4d:79:  2c:4a
```

8.6.1.7 Processing of the CHALLENGE request message

The Power Transmitter Product parses the CHALLENGE request message to recover the slot index then composes the full “TBSAuth” data (the ‘message’ that is to be signed) as defined in Table 20 (in Section 5.1). The assembled “TBSAuth” data for this challenge is shown in Table 30.

Table 30: TBSAuth data - example 1

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x41 (Prefix 'A')							
B_1 to B_{32}	0x4629653ad1ceb37c6a36f0cc11b4291686392785f0f826dfded35eac5fcc50fc (Certificate Chain Hash)							
B_{33} to B_{50}	0x1b003c2f44d265e340a5ccac0f724d792c4a (Challenge Request)							
B_{51} to B_{53}	0x1311fc (Copy of B0 to B2 of a CHALLENGE_AUTH response)							

The concatenated hex form of this TBSAuth data is:

```
41:46:29:65:3a:d1:ce:b3:7c:6a:36:f0:cc:11:b4:29:  16:86:39:27:85:f0:f8:26:df:de:d3:5e:ac:5f:cc:50:
fc:1b:00:3c:2f:44:d2:65:e3:40:a5:cc:ac:0f:72:4d:  79:2c:4a:13:11:fc
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

The Power Transmitter Product then computes the SHA256 hash of the “TBSAuth” data:

```
0x4c1e51e60cab65107e212293a7586d5b87e7cd05a2ea43dda331f4779ae8e5bb
```

and then securely generates an ECC signature using its private key. The resulting signature (*r*,*s*) values are shown here, together will all the values required to re-generate them - including the secure random number (*k*) and the SHA256 hash “digest” of the “TBSAuth” data (*z*):

signature details

```
z = 0x4c1e51e60cab65107e212293a7586d5b87e7cd05a2ea43dda331f4779ae8e5bb # message (digest)
r = 0x9159946fe34c196a8d7a1eb50c4e06e653688d08dd8722dbe7462a637b06c5ab # signature.r
s = 0xec3e9c8f8c773115b5861308b6b8ea9901cb66f6f2f8e110df697559535feb81 # signature.s
```

8.6.1.8 CHALLENGE_AUTH response

The Power Transmitter Product then composes a CHALLENGE_AUTH response message including the signature data, as shown inTable 31 (composed as defined by Table 19 (in Section 5.1)).

Table 31: CHALLENGE_AUTH response message — example 1

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x13 (CHALLENGE_AUTH Message Header)							
B_1	0x1 (Maximum Auth Protocol Version)				0x1 (Slots Populated Mask)			
B_2	0xfc (Certificate Chain Hash LSB)							
B_3…B_{34}	0x91:59:94:6f:e3:4c:19:6a:8d:7a:1e:b5:0c:4e:06:e6: 53:68:8d:08:dd:87:22:db:e7:46:2a:63:7b:06:c5:ab (Signature *r* value)							
B_{35}…B_{66}	0xec:3e:9c:8f:8c:77:31:15:b5:86:13:08:b6:b8:ea:99: 01:cb:66:f6:f2:f8:e1:10:df:69:75:59:53:5f:eb:81 (Signature *s* value)							

The concatenated hex form of this response is:

```
13:11:fc:91:59:94:6f:e3:4c:19:6a:8d:7a:1e:b5:0c:  4e:06:e6:53:68:8d:08:dd:87:22:db:e7:46:2a:63:7b:
06:c5:ab:ec:3e:9c:8f:8c:77:31:15:b5:86:13:08:b6:  b8:ea:99:01:cb:66:f6:f2:f8:e1:10:df:69:75:59:53:
5f:eb:81
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

8.6.1.9 Processing the CHALLENGE_AUTH response message

On receipt of the CHALLENGE_AUTH response, the Power Receiver Product parses the data and assembles what should be a facsimile of the "TBSAuth" bytes that the Power Transmitter Product signed (using the definition given in Table 20 (in Section 5.1)).

```
41:46:29:65:3a:d1:ce:b3:7c:6a:36:f0:cc:11:b4:29:  16:86:39:27:85:f0:f8:26:df:de:d3:5e:ac:5f:cc:50:
fc:1b:00:3c:2f:44:d2:65:e3:40:a5:cc:ac:0f:72:4d:  79:2c:4a:13:11:fc
```

The Power Receiver then calculates the SHA256 hash of this data:

```
0x4c1e51e60cab65107e212293a7586d5b87e7cd05a2ea43dda331f4779ae8e5bb
```

signature verification details

The Power Receiver Product then verifies that the signature data returned in the CHALLENGE_AUTH response is consistent with the private key associated with the public key of the Product Unit Certificate in slot 0. The full list of inputs required to be fed into an ECC signature verification routine to test this signature are summarized here:.

```
z = 0x4c1e51e60cab65107e212293a7586d5b87e7cd05a2ea43dda331f4779ae8e5bb # message hash (digest)
x = 0x77b1f30e5d79a63fbcc35de8436e45d89c15f9998e8b8f2c6001caedae5f859  # public_key.x
y = 0x3a5076d2c7a4af0bc56b479de16ada110c0aefd739e1f04d0dd763eb9321353 # public_key.y
r = 0x9159946fe34c196a8d7a1eb50c4e06e653688d08dd8722dae7462a637b06c5ab # signature.r (to be verified)
s = 0xec3e9c8f8c773115b5861308b6b8ea9901cb66f6f2f86110df697559535feb81 # signature.s (to be verified)
```

After successfully verifying the signature, the Power Receiver Product considers that the Power Transmitter Product has passed the authentication test.

8.6.2 Test of example 2, typical length Certificate Chain

This authentication example tests the Certificate Chain we created in Section 8.5.2, *Example 2: typical length Certificate Chain*. In addition, it also differs from the first example in terms of flow, as the Power Receiver Product breaks the "certificate request" process into two steps.

8.6.2.1 GET_DIGESTS request

Once the Power Receiver Product has confirmed that the Power Transmitter Product supports authentication, the exchange of messages begins with a GET_DIGESTS request from the Power Receiver Product. The Power Receiver Product chooses to ask for the digests of all slots and queries Power Transmitter Product for chain digests by sending the GET_DIGESTS request message detailed in Table 32(composed following the definition in Table 13 (in Section 5.1))

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 32: GET_DIGESTS request message — example 1

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x19 (GET_DIGESTS Message Header)							
B_1	0x0 (Reserved)				0xF (Slot Mask)			

The concatenated hex form of this is:

```
19:0f
```

8.6.2.2 DIGESTS response

The Power Transmitter Product returns a DIGESTS response message composed according to the definition given in Table 17 (in Section 5.1).

As the Power Transmitter Product only has a single Certificate Chain (populated in slot 0) it sets both slot masks of the message to 0x1 and includes the 32-byte SHA-256 digest of its slot 0 Certificate Chain as the 'digests returned' data (i.e. the digest value given in Table 8.5.1). This message is shown in Table 33

Table 33: DIGESTS response

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x11 (DIGESTS Message Header)							
B_1	0x1 (Slots Populated Mask)				0x1 (Slots Returned Mask)			
B_2 to B_{33}	0xe36b91faf190e0874ce8656c28e23376e8ba29bdfbcfaf3cb34a81dd48847f22 (Digests Returned)							

The concatenated hex form of this DIGESTS response message is:

```
11:11:e3:6b:91:fa:f1:90:e0:87:4c:e8:65:6c:28:e2:  33:76:e8:ba:29:bd:fb:cf:af:3c:b3:4a:81:dd:48:84:
7f:22
```

8.6.2.3 GET_CERTIFICATE request

Not recognizing the digest data returned in the last message, the Power Receiver Product needs to retrieve a copy of the full Certificate Chain from the Power Transmitter Product. Rather than asking for the entire Certificate Chain in one request, the Power Receiver Product starts by asking the Power Transmitter Product for the first 512 bytes of the Certificate Chain in slot 0 by sending the GET_CERTIFICATE request described in Table 34 (composed as defined by Table 14).

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025

Table 34: GET_CERTIFICATE request message – example 2

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
B_0	0x1a (GET_CERTIFICATE Message Header)							
B_1	0x0 (OffsetA8)			0x2 (LengthA8)			0x0 (Slot Number)	
B_2	0x00 (Offset70)							
B_3	0x00 (Length70)							

The concatenated hex form of this is:

```
1a:08:00:00
```

8.6.2.4 First CERTIFICATE response

The Power Transmitter Product responds with a CERTIFICATE response message that includes the first 512 bytes of the Certificate Chain in slot 0 composed as defined in Table 18 (in Section 5.1).

On receipt of this GET_CERTIFICATE request, the Power Transmitter responds with a CERTIFICATE response using the format defined in Table 18 (in Section 5.1). As the request is for the full slot 0 Certificate Chain, the message is simply composed of the CERTIFICATE Message Header byte (0x12) followed by the bytes of the entire Certificate Chain held in slot 0. The concatenated hex form of this response is:

```
12:02:91:cb:29:05:19:c6:52:67:94:c2:4d:d5:8b:cd:   15:c2:0f:39:96:a8:ac:62:b2:8f:75:91:44:46:77:b3:
9c:0a:9c:30:82:01:49:30:81:ef:a0:03:02:01:02:02:   0a:00:ef:f1:1d:6a:02:fe:a1:8f:d2:30:0a:06:08:2a:
86:48:ce:3d:04:03:02:30:11:31:0f:30:0d:06:03:55:   04:03:0c:06:57:50:43:43:41:58:30:22:18:0f:32:30:
30:30:30:31:30:31:30:30:30:30:30:30:5a:18:0f:39:   39:39:39:31:32:33:31:32:33:35:39:35:39:5a:30:12:
31:10:30:0e:06:03:55:04:03:0c:07:43:41:43:41:2d:   58:31:30:59:30:13:06:07:2a:86:48:ce:3d:02:01:06:
08:2a:86:48:ce:3d:03:01:07:03:42:00:04:b3:d1:ba:   7d:2e:61:4f:f7:8c:d6:63:fa:50:70:bc:ef:ba:52:3c:
9d:3a:b1:c8:f0:c5:da:6c:f5:03:de:43:e2:9c:60:30:   50:ec:4b:25:3e:ae:64:0b:cc:38:18:0a:ec:f8:35:c9:
4f:0b:d8:02:92:11:b5:bb:7a:a7:fa:85:6f:a3:2a:30:   28:30:12:06:03:55:1d:13:01:01:ff:04:08:30:06:01:
01:ff:02:01:00:30:12:06:05:67:81:14:01:01:01:01:   ff:04:06:04:04:00:00:00:01:30:0a:06:08:2a:86:48:
ce:3d:04:03:02:03:49:00:30:46:02:21:00:90:6e:b5:   64:05:aa:b1:b1:2e:af:f0:da:92:70:65:e6:df:5d:37:
06:84:45:95:71:70:71:97:58:e9:a9:6c:da:02:21:00:   f8:8a:2c:c4:d8:c1:b1:b6:9a:60:2d:f5:16:8a:7a:34:
e4:8f:31:3a:2a:48:4c:e6:63:62:07:1a:60:e8:95:29:   30:82:01:1e:30:81:c5:a0:03:02:01:02:02:09:00:f9:
d5:c4:77:48:c4:eb:11:30:0a:06:08:2a:86:48:ce:3d:   04:03:02:30:12:31:10:30:0e:06:03:55:04:03:0c:07:
43:41:43:41:2d:58:31:30:22:18:0f:32:30:32:31:30:   38:32:34:31:32:30:36:30:30:5a:18:0f:32:30:32:31:
30:38:32:35:31:32:30:36:30:30:5a:30:18:31:16:30:   14:06:03:55:04:03:0c:0d:30:30:30:30:34:32:2d:4d:
6f:64:65:6c:33:30:39:30:13:06:07:2a:86:48:ce:3d:   02:01:06:08:2a:86:48:ce:3d:03:01:07:03:22:00:02:
ef
```

IECNORM.COM : Click to view the full PDF of IEC 63563-9:2025