ETSI TS 103 597-2 V1.1.1 (2021-04)

Methods for Testing and Specification (MTS);
Test Specification for MQTT;
Part 2: Security Tests

Reference
DTS/MTS-TSTMQTT-2

Keywords
security, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights 4
Foreword 4
Modal verbs terminology 4
Introduction 4
1 Scope 6
2 References 6
2.1 Normative references 6
2.2 Informative references 6
3 Definition of terms, symbols and abbreviations 7
3.1 Terms 7
3.2 Symbols 7
3.3 Abbreviations 7
4 Security Test Objectives 7
5 Security testing techniques (preliminary consideration) 8
5.1 Fuzz Testing 8
5.1.1 General Description 8
5.1.2 Example Approach 8
5.1.3 MQTT-specific Considerations 9
5.2 Penetration Testing 10
5.3 Testing for vulnerabilities 10
5.4 Further approaches 10
6 Test Configurations 11
7 MQTT Security Test Purposes 11
7.1 Introduction 11
7.2 Authentication/authorization 12
7.3 Encryptions 13
7.4 Specific protocol security considerations 13
8 Vulnerability Test Samples 13
Annex A (informative): Sample security test catalogue 15
A.1 Introduction 15
History 16

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

The present document is part 2 of a multi-part deliverable. Full details of the entire series can be found in part 1 [2].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

The present document provides an introduction and guide for developers and users investigating in security testing of the MQTT communication protocol. It will be a reference base for both client side test campaigns and server side test campaigns addressing the security issues. It belongs to a multipart technical specification addressing the most relevant testing aspects of MQTT:

- conformance;
- security; and
- performance testing.

While the conformance testing part presents a complete set of test purposes, the content for security and performance parts is different and focus on evaluating relevant testing techniques and the provision of samples that are specific for MQTT. For this reason, the structure of the document consists of four main clauses: the first two clauses address the security test objectives, techniques and methods to be considered for MQTT. Concrete practical hints and samples and configuration notes are provided where feasible. The latter two clauses focus on the security mechanisms and implementation notes mentioned in the MQTT protocol standard and security vulnerabilities known from relevant vulnerability databases. Concrete test purposes have been described using the Test Description Language (TDL) standardized by ETSI.

1 Scope

The present document provides general security considerations and guidelines about the Message Queuing Telemetry Transport (MQTT) protocol. The collective ideas presented in the present document are enriched with example Test Purposes (TPs) to outline possible implementations.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] OASIS Standard: "MQTT Version 3.1.1".

[2] ETSI TS 103 597-1: "Methods for Testing and Specification (MTS); Test Specification for MQTT; Part 1: Conformance Tests".

[3] ETSI ES 203 119-4: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Eclipse IoT-Testware v.0.1.0.

NOTE: Available at https://projects.eclipse.org/projects/technology.iottestware.

[i.2] ETSI ES 202 951: "Methods for Testing and Specification (MTS); Model-Based Testing (MBT); Requirements for Modelling Notations".

[i.3] ETSI TS 103 646: "Methods for Testing and Specification (MTS); Test specification for foundational Security IoT-Profile".

[i.4] IEC 62443-4-2: "Security for industrial automation and control systems. Technical security requirements for IACS components".

[i.5] CVE-2019-5432.

NOTE: Available at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5432.

[i.6] CVE-2018-12543.

NOTE: Available at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12543.

[i.7] CVE-2019-11779.

NOTE: Available at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11779.

[i.8] ETSI TR 101 583: "Methods for Testing and Specification (MTS); Security Testing; Basic Terminology".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

black-box testing: testing activity conducted without knowledge of the internal structure of the system under test

grey-box testing: testing activity conducted with a partial knowledge of the internal structure of a system under test

system under test: real open system in which the implementation under test resides

NOTE: See ETSI ES 202 951 [i.2].

white-box testing: testing based on an analysis of the internal structure of the component or system under test

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CVE	Common Vulnerabilities and Exposures
DoS	Denial of Service
IUT	Implementation Under Test
JSON	JavaScript Object Notation
MITM	Man-in-the-Middle
MQTT	MQ Telemetry Transport
SQL	Structured Query Language
SUT	System Under Test
TDL	Test Description Language
TDL-TO	Test Description Language - Test Objectives
TLS	Transport Layer Security
TSS	Test Suite Structure
UTF	UCS Transformation Formats
XOR	Exclusive OR

4 Security Test Objectives

When talking about security test objectives mean assets that are worth protecting. This clause does not focus on how to protect those assets but raising awareness when it comes to implementing the protocol, especially within an IoT environment. Of course, the following list does not claim to be complete. Prior to this, all environmental conditions, such as the domain and location, should be clarified beforehand.

Integrity in the present document is more related to data integrity. It should be possible to answer questions like: Can the data be trusted? Do the data have integrity? Were the data transmitted without manipulation?

Availability refers to the requirement that the system available in general. DoS attacks should not lead to an unavailable system. Unusual setbacks for system performance are not expected because the system should operate at least with basic functionalities.

Robustness refers to the ability to be resilient against (unexpected) situations like receiving malformed data or communication flows with correct data. Robustness and Availability are closely related. In addition, performance considerations are related to robustness because of the mere amount of input data.

5 Security testing techniques (preliminary consideration)

5.1 Fuzz Testing

5.1.1 General Description

Fuzzing is an effective negative black box testing method of finding unknown vulnerabilities in software. A System Under Test (SUT) is exposed via its interfaces to unexpected data. The idea is to send partially invalid data to get the system into an unexpected state. Inputs are generated randomly or systematically by mutating valid data or creating new data according to specifications. Most of the input is rejected because of internal validation mechanisms of the system. Those rejected inputs are considered ineffective since their execution does not lead to the possibility of exposing weakness which reduces the overall effectiveness of a fuzzing campaign. Model-based security testing does target this issue by generating test cases according to the systems model.

5.1.2 Example Approach

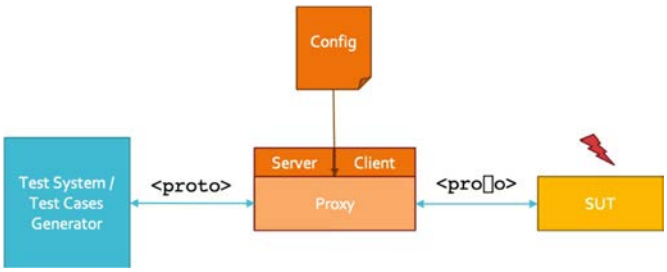

Figure 1: Fuzzing configuration

One possible application of the fuzzing approach can be found in the Eclipse IoT-Testware project [i.1] that implement a fuzzing proxy. The Fuzzing Proxy is a MITM (Man-in-the-middle) Fuzzer which is capable of proxying the network traffic between two systems and altering this traffic on behalf of predefined rules. The Fuzzing Proxy does not generate any message on its own. To trigger the procedure, (more or less) valid templates need to be provided.

The approach follows a 5-step fuzzing workflow (see figure 2) that is described in the following.

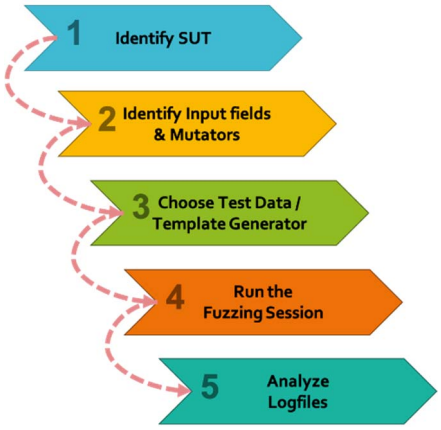

Figure 2: Fuzzing workflow

Step 1 Identify the SUT and step 2 Identify input fields are characterized together. The first step is to identify if the SUT is in the correct scope where objectives with fuzzing will be achieved. Identifying input fields and corresponding mutators for the fuzzing session is probably the most challenging part in the whole workflow. In the referred example (see Eclipse IoT-Testware) interesting input fields can be chosen and corresponding mutators defined in the configuration file. The configuration file provided to the Fuzzing Proxy contains abstract fuzzing instructions which are used at runtime for manipulation of proxied messages. The configuration file is a plain JSON file following a specific schema.

The step 3 is to choose a test data generator. As stated above, the fuzzing solution does not generate any test data but manipulate given test data. Therefore a test data generator is needed. This can be a simple client connecting to the SUT and sending request messages or other test solutions like the ones provided by the IoT-Testware.

Step 4 is the actual fuzzing by means of manipulating the incoming test data. Concepts used here are mainly mutators and filter. Fuzzing Mutators are one of the basic concepts of the Fuzzing Proxy. It mutates (changes) the input based on different rules. In other words, unary or binary operators are applied to change the incoming message. Unary Operators, like NOT, as the name implies, are unary with respect to the number of parameters which they expect. An unary operator expects only one single parameter. It takes the value of the specified field (as the one and only parameter) and applies a fuzzing operation on it. Binary Operators on the other hand, like XOR, take two parameters, the value of the specified field and either a fixed value or a generator as the second parameter. Next to mutators, filters are the second building blocks on the path of building fuzzing rules. These fuzzing filters are conceptually very similar to Wireshark's DisplayFilters and serve pretty much the same purpose. As one might want to intercept more complex protocol behaviours, altering each single message would be a bad idea. The concept of filters allows the user to pick only specific messages for fuzzing, while other messages not matching any filter are simply passed through without being fuzzed.

The last step is to analyse the fuzzing logging. By having the log information, further evaluate potential flaws or precise protocol violations can be checked.

5.1.3 MQTT-specific Considerations

Regarding MQTT, different message fields can be considered for the fuzzing approach. The fixed-size 2-byte header can be started with. Exhausting non-defined or reserved values within the possible range opens various possibilities to expose potential vulnerabilities. The following fields are defined optional containing variable length header and message payload information. Furthermore, it is essential to know the context of the application or device and put it into consideration. For example, if there is a database behind the MQTT server, SQL injections can be infiltrate into the system via the MQTT payload. This might be far-fetched but also simple content transmitted within the payload can cause unwanted behaviour of the system.

5.2 Penetration Testing

A penetration test is a special kind of test where an attack on a system or network is simulated by an attacker or attacker team. The attacker attempts to break into the system and take control. In contrast to testing during development, no parts or artifacts of the system to be tested are checked and there are no functional tests. The system to be tested is often the finished system as it is used in production environment. Penetration testing can be done as black-box test or white-box test and all in between. A real-world attacker usually has no information about the system he wants to penetrate. That is why a black box penetration test is the closest thing to a real attacker. But with additionally information's about the SUT (e.g. white-box testing or grey-box testing) a penetration tester can reduce the effort of the complete penetration test or raise the quality of the result.

The approach of penetration testing often follows five phases:

1) Planning and reconnaissance
2) Scanning
3) Gaining access
4) Maintaining access
5) Analysis

5.3 Testing for vulnerabilities

Testing for vulnerabilities is an approach, where already known vulnerabilities from other systems are used to check the SUT. These vulnerabilities can be found in databases and bug reports in the internet, e.g. in CVE databases. There is always a good chance, that common mistakes can be found in different implementations or that an implementation uses an older library where this error still exists.

Already found vulnerabilities and exploits building upon this are fundamental for penetration testing, clause 5.2.

Clause 8 provides some examples of specific Test Purposes that refer to real world vulnerabilities that was found in systems using MQTT protocol.

5.4 Further approaches

The security testing methods and techniques continuously grow and evolve following new attack strategies and pattern. Basic security techniques as presented above are well-known from ETSI TR 101 583 [i.8] as new security testing approaches (e.g. address spoofing and amplification attacks). Security testers always need to be aware and check latest results from research and practical experience reports.

6 Test Configurations

The security test configurations are derived from the SUT access points and functional test configurations. For all test configurations presented in this clause, a sniffing tool like Wireshark is recommended, but not shown in figure 3.

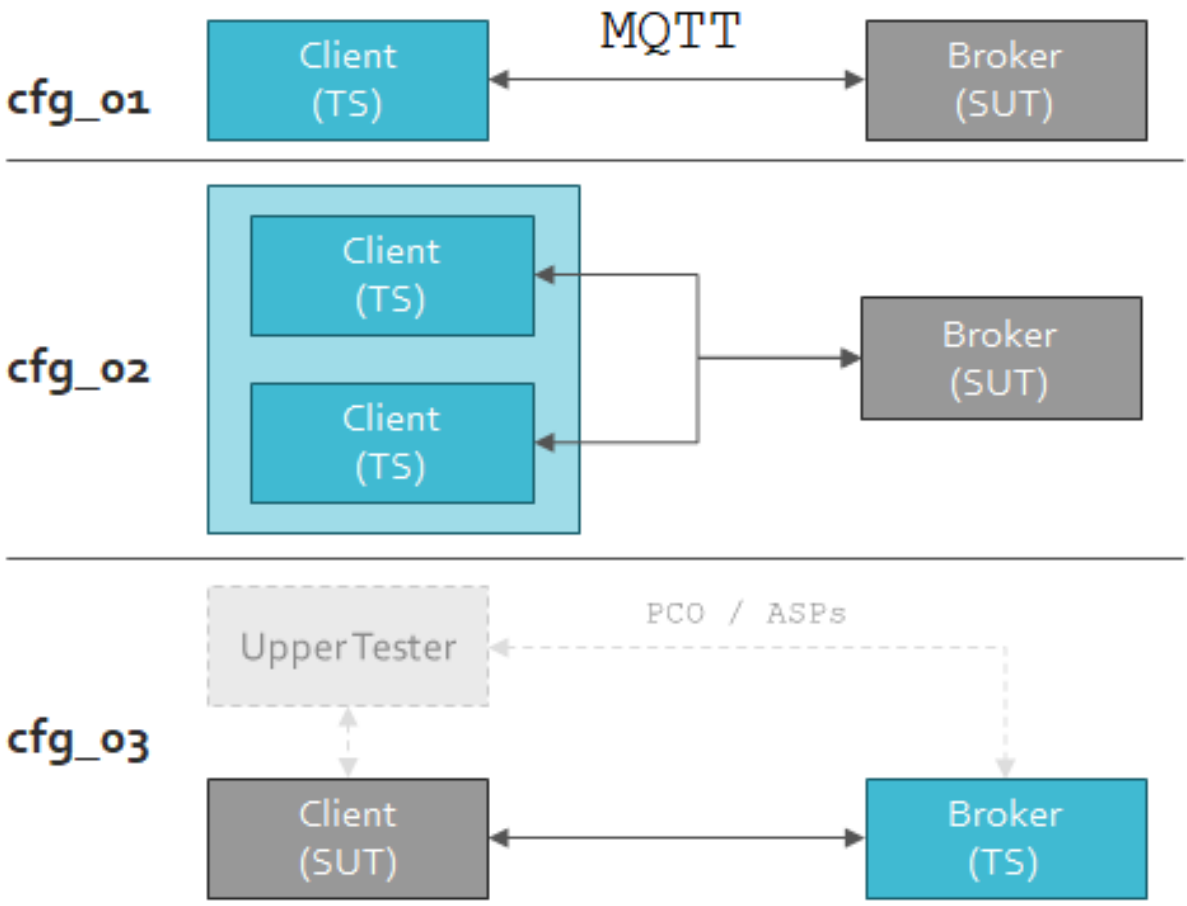

Figure 3: MQTT test configurations

7 MQTT Security Test Purposes

7.1 Introduction

Several TPs can be derived from the security test objectives and testing techniques mentioned in clauses 4 and 5. Some important aspects for MQTT security testing include:

- Robustness (coverage criteria, test suite execution time, number of test cases to be executed, number of test data):
  - Data level: malformed packets/data (e.g. MQTT length fields), encoding UTF-8;

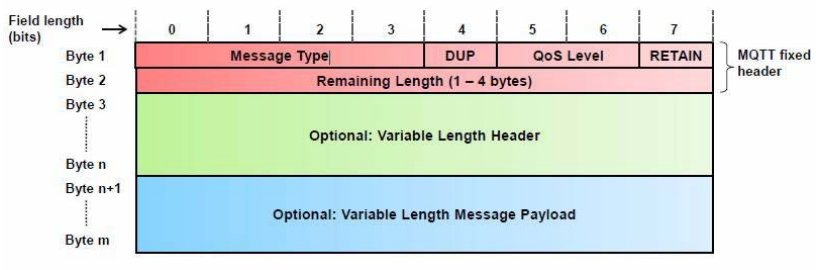

Figure 4: MQTT message format

- behaviour level: MQTT scenarios.

- DoS (availability).
- Penetration testing ideas:
  - apply brute force attacks to authentication procedure: using CONNECT message.
- Spoofing.
- Insecure protocol configurations:
  - which parameters lead to insecure configurations of the protocol, e.g. fallback to insecure cryptographic protocols, key length, etc.

Further ideas can be derived from TPs related to the IEC 62443-4-2 [i.4] security profile testing (ETSI TS 103 646 [i.3]), see examples below.

Following the MQTT TSS and TP naming conventions introduced in ETSI TS 103 597-1 [2], clause 4, security TPs are numbered, starting at 001, within the main scope "SEC" (security). The identifiers for the second level scope needs to be derived from the target security requirement or security test methods, e.g. "AUT" for Authentication/authorization and "CVE" for TPs derived from public vulnerability data bases.

7.2 Authentication/authorization

The following TP has been derived from TP_CR_1_1_Identification_authentication_1 in ETSI TS 103 646 [i.3].

TP Id	TP_ MQTT_BROKER_SEC_AUT_001
Test Objective	Ensure the IUT identifies and authenticates users. Case invalid account identifier/invalid authenticator
Reference	MQTT 3.2.2.3
PICS Selection	
Initial Conditions	

with {
the IUT being_in the initial_state
}	

Expected Behaviour	

ensure that {
when {
the IUT receives CONNECT message with the credentials
account identifier indicating value "invalid account identifier",
account authenticator indicating value "invalid account authenticator";
}
then {
the IUT sends "Connect Return Code 0x02 Connection Refused, identifier rejected" }
}	

Final Conditions	

7.3 Encryptions

As mentioned in clause 5.4.5 of the protocol definition [1] encryption may be provided using e.g. TLS or VPN.

7.4 Specific protocol security considerations

MQTT specific security considerations have been provided and discussed in clause 5.4.11 of the protocol definition [1].

8 Vulnerability Test Samples

Testing for Vulnerabilities can be done using vulnerability data bases as inspiration for test cases, such as using associated weaknesses from CVE.

Sample TPs from CVEs [i.5], [i.6] and [i.7] that were found in existing MQTT implementations are the following:

TP Id	TP_MQTT_BROKER_SEC_CVE_001
Test Objective	Verify that the IUT does not crash on reception of a malformed SUBSCRIBE Control Packet.
Reference	CVE-2019-5432 [i.5]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the PROBE_CLIENT established the MQTT_CONNECTION containing
keep_alive corresponding to PX_PROBE_CON_KEEP_ALIVE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
payload containing
filterLength corresponding to TOPIC_FILTER_LEN_SEC_CVE_01,
topic_filter corresponding to TOPIC_FILTER_LEN_SEC_CVE_01;;
from the ATTACKER_CLIENT
}
then {
the IUT not closes the TCP_CONNECTION for the PROBE_CLIENT
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SEC_CVE_002
Test Objective	Verify that the IUT does not crash on reception of a malformed PUBLISH Control Packet having topic names prefixed with '$SYS'.
Reference	CVE-2018-12543 [i.6]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the ATTACKER_CLIENT having a MQTT_CONNECTION to the IUT
and
the PROBE_CLIENT established the MQTT_CONNECTION containing
keep_alive corresponding to PX_PROBE_CON_KEEP_ALIVE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_SEC_CVE_02;
from the ATTACKER_CLIENT
}
then {
the IUT not closes the TCP_CONNECTION for the PROBE_CLIENT
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SEC_CVE_003
Test Objective	Verify that the IUT does not crash on reception of a SUBSCRIBE Control Packet containing a topic that consists of approximately 65400 topic hierarchy separators ('/' character).
Reference	CVE-2019-11779 [i.7]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the ATTACKER_CLIENT having a MQTT_CONNECTION to the IUT
and
the PROBE_CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
payload containing
topic_filter corresponding to TOPIC_FILTER_SEC_CVE_03;;
from the ATTACKER_CLIENT
}
then {
the IUT not closes the TCP_CONNECTION for the PROBE_CLIENT
}
}	

Final Conditions	

Annex A (informative): Sample security test catalogue

A.1 Introduction

Test purposes presented in the present document have been produced using the Test Description Language (TDL-TO) according to ETSI ES 203 119-4 [3]. The TDL-TO library modules corresponding to the Test purpose catalogue are contained in archive ts_10359702v010101p0.zip which accompanies the present document.

History

Document history		
V1.1.1	April 2021	Publication