TRUSTWORTHY SOFTWARE

SPECIFICATION DOCUMENT

TRUSTWORTHY SOFTWARE ESSENTIALS (TSE)

February 2016
ISSUE 1.2 – TLP WHITE

TS502-0

DOCUMENT CONTROL

CHANGE RECORD

VERSION	DATE	SUMMARY
0.A	23/06/2015	First draft of document, for internal TSI comment
0.B	24/06/2015	Updated with internal comments
0.C	30/06/2015	Minor update and formatting
0.D	15/07/2015	Minor update, including update of naming conventions
0.E	10/07/2015	Updated with internal/stakeholder comments
1.0	02/12/2015	Updated with final corrections
1.1	07/12/2015	Updated with stakeholder comments
1.2	16/02/2016	Updated to include reference to fundamental requirements

DOCUMENT INFORMATION

DOCUMENT ID	TSI/2014/206
PRODUCTION ID	N/A
TSK-ID	TS502-0
PUBLISHER	UK-TSI, IMC, Westwood Heath, CV4 7AL
TRAFFIC LIGHT PROTOCOL (TLP)	WHITE: Open Published Material with no restrictions on distribution. *NB. Please see Glossary for further information*
RIGHTS.COPYRIGHT:	
RIGHTS.CUSTODIAN:	TSI Programme Support Officer (+44 300 030 1928 / enquiries@uk-tsi.org)

FOREWORD

TRUSTWORTHY SOFTWARE INITIATIVE

The Trustworthy Software Initiative (TSI) is part of the UK Government's National Cyber Security Programme to improve the UK's ability to combat cyber risks and ensure that the UK leads the way in trustworthy software systems and expertise.

The objective of TSI is to provide the knowledge, skills and capability for supply, demand and "corpus" (education and research) communities such that trustworthy software can be designed, implemented, sustainably maintained and assured in a risk-based, whole-life process.

TSI works with organisations and individuals in the UK, and international partners, including government, academia, private/public companies, software developers and users, to achieve a recognised level of trust of software by providing targeted education, skills, standards and guidance.

LICENCE CONDITIONS

DISCLAIMER

FURTHER INFORMATION

For further information relating to other aspects of Trustworthy Software, including the more comprehensive Trustworthy Software Framework, please visit: www.uk-tsi.org.

For all associated and/or supporting documents, please see 'Annex A: References' of this document.

CONTENTS

1 INTRODUCTION

1.1. TRUSTWORTHY SOFTWARE

OVERVIEW

With our daily lives and industrial processes becoming increasingly reliant on a wide range of underpinning software, there is a pressing need to address the quality and robustness of that software in order to establish its "trustworthiness" and therefore ensure that it performs as it should, when it should and how it should.

This means addressing the trustworthiness of software throughout its life-cycle, from development through to disposal.

FACETS OF TRUSTWORTHINESS

For this specification, the Trustworthy Software Initiative (TSI) identifies trustworthiness to predominantly consist of the following five facets:

Safety, Reliability, Availability, Resilience and Security

WHEN IS SOFTWARE TRUSTWORTHY?

No software asset can be proven, or even be expected, to be completely free of all defects i.e. free from conditions which could cause it to fail or behave in an unexpected manner. However, it should have a level of trustworthiness commensurate to the purpose for which it is used.

In order to determine the appropriate level of trustworthiness for a software asset, TSI recommends a risk-based approach to determining the level of trustworthiness required, whereby the reliance on the software to provide trustworthiness is considered together with the maximum impact of a defect/deviation in the software.

TRUSTWORTHY SOFTWARE CONTROLS

The following table shows the two sets of controls that should be adopted, dependent upon the Trustworthiness Level (TL) required, in order that an appropriate level of trustworthiness is achieved:

TL	SOFTWARE AUDIENCE	CONTROL SET
TL 0	No requirement for Trustworthy Software	No Requirement
TL 1	Mass Market with Implicit Need (M/I)	TS Essentials (TSE)
TL 2		Baseline TS controls forming a sub-set of the TS Framework (TSF)
TL 3	Mass Market with Explicit Need (M/E)	TS Framework (TSF)
TL 4	Niche with Explicit Need (N/E)	Comprehensive TS controls utilising the full TS Framework (TSF)

1.2. PURPOSE AND APPLICABILITY OF DOCUMENT

This document provides a specification for the Trustworthy Software Essential (TSE) controls that should be adopted by organisations in order to ensure the trustworthiness of the software they produce, procure or use.

It is intended as essential reading for those in the organisation responsible for implementing the decision to adopt trustworthy software, and is applicable to organisations of all sizes who either:

- have a basic requirement for trustworthy software, or
- require a baseline Trustworthiness Level (TL) of 1 or 2.

1.3. USING THIS DOCUMENT

This document sets out the baseline requirements for managing trustworthiness during the software life-cycle (including Governance, Responsibilities and the Documentation required), together with the controls comprising Trustworthy Software Essentials.

The TSE controls are organised under 5 **Essential (E) Objectives**:

Scope for Use (E1)

Coding Practices (E2)

Use Tools Effectively (E3)

Defect Management (E4)

Artefact Management (E5)

The TSE controls are a subset of the comprehensive Trustworthy Software Framework (TSF) techniques and therefore each have a unique control serial number which cross-references back to the TSF. This is stated at the end of each control in the format of [xx.nn.nn], where xx denotes:

GV Governance controls

RI Risk controls

PE Personnel controls

PH Physical controls

PR Procedural controls

TE Technical controls

CM Compliance controls

It is intended that each control listed should be implemented in a prescriptive manner, as it is only by full adoption that the majority of the most commonly encountered risks can be mitigated.

In addition to these baseline requirements, further fundamental control requirements (considered as the normative approach to ensuring software trustworthiness within PAS 754) are identified for each Essential Objective (E1-E5). These fundamental controls are required to be established in addition to the basic controls listed under each objective.

2 MANAGING TRUSTWORTHINESS

2.1. GOVERNANCE

A Trustworthy Software Management System (TSMS) should be established in order to document the way in which software trustworthiness is governed within your organisation. The TSMS should encompass the policies and mechanisms required to ensure that an appropriate level of trustworthiness is achieved and continues to be maintained throughout the software life-cycle.

Where appropriate, the TSMS may be incorporated into existing management systems (for example an ISMS or QAMS) and/or certification schemes.

2.2. ROLES & RESPONSIBILITIES

The following roles should be established as a minimum and, where appropriate, may be encompassed into an existing role(s):

THE BOARD

The organisation's Board should be responsible for the overall governance of the provision of Trustworthy Software and, in particular, should be fully cognisant of the organisation's key information assets and the organisational impact in the event of compromise, loss or unavailability.

TRUSTWORTHY SOFTWARE RELEASE AUTHORITY (TSRA)

The TSRA should have day-to-day responsibility for the provision of Trustworthy Software and should be empowered by the Board with sufficient authority to fulfil this role.

2.3. DOCUMENTATION

The following documentation should be established as a minimum set of work products and be used to provide an evidence trail as necessary. Where appropriate, these may be incorporated into existing documents:

TRUSTWORTHY SOFTWARE DEFECT AND DEVIATION LIST (TSDDL)

The TSDDL should be used to record all defects and deviations both during Realisation (R-TSDDL) and In-service (I-TSDDL).

TRUSTWORTHY SOFTWARE RELEASE NOTICE (TSRN)

The Trustworthy Software Release Notice (TSRN) should document the way in which the principles of trustworthiness have been addressed during the specification and realisation of the software asset, together with all constraints, dependencies and unmitigated defect/deviations.

TRUSTWORTHY SOFTWARE CONSTRAINT AND DEPENDENCY MODEL (TSCDM)

The Trustworthy Software Constraint and Dependency model (TSCDM) should document the way in which all elements of the software asset are intended to be deployed, configured and operated.

3 TS ESSENTIAL CONTROLS

3.1. SCOPE FOR USE (E1)

OBJECTIVES

The objectives of 'Scope for Use' are to fully understand the requirements of the software you are producing or procuring, in particular: the environment in which it will operate, the functionality it will have and the requirements it needs to fulfil.

FUNDAMENTAL CONTROLS

The following control measures are required to be established (or demonstrated) to meet the TSE objective 'Scope for Use' (E1):

- TSMS (section 2.1)
- TSRA (section 2.2)
- TSCDM (section 2.3)

BASIC CONTROLS

The controls listed below meet the TSE objective 'Scope for Use' (E1) and each should be implemented in a prescriptive manner as required by the TSMS.

UNDERSTAND REQUIREMENTS [PR.03]:

- **Specify Explicit / Functional Requirements [PR.03.10]**
  You should define what the software is supposed to do, as in its specific behaviour and/or functions. This should be recorded in the Software Requirement's Specification (SRS) and/or TSCDM as appropriate.
- **Specify Implicit / Non-functional Requirements [PR.03.20]**
  You should define the way in which the system operates, to include specific requirements relating to Safety, Reliability, Availability, Resilience, Security, Usability and Performance. This should be recorded in the SRS and/or TSCDM as appropriate.
- **Understand Implicit / Non-objective Requirements [PR.03.30]**
  You should define the business requirements of the software and any constraints imposed on the software, either by the business or by other systems. This should be recorded in the SRS and/or TSCDM as appropriate.
- **Understand Use Cases [PR.03.40]**
  You should understand how the user will interact with the software in order to achieve specific goals and focus on what the software is required to do rather than how it will do it. This should be recorded in the SRS and/or TSCDM as appropriate.
- **Monitor and record Derived Requirements [PR.03.50]**
  You should document all additional requirements that have been identified as a result of realisation activities (and have not already been defined as part of PR.03.10, PR.03.20 and/or PR.03.30) in the TSCDM as by definition, they should not be recorded in the SRS.

SEEK TRUSTWORTHY REALISATION [TE.05]:

- **Implement only the minimum set *of features* of users to meet requirements [TE.05.80]**
  You should ensure that the functionality of the system is sufficient to meet the requirements of the user, without containing extraneous features.

3.2. CODING APPROACH (E2)

OBJECTIVES

The objectives of 'Coding Approach' are to ensure that the software you produce or procure is structurally sound, such that not only does it perform 'as expected', 'when expected' and 'how expected', it is also facilitates maintenance, debugging and enhancement.

FUNDAMENTAL CONTROLS

The following control measures are required to be established (or demonstrated) to meet the TSE objective 'Coding Approach' (E2):

- TSMS (section 2.1)
- TSDDL (section 2.3)

BASIC CONTROLS

The controls listed below meet the TSE objective 'Coding Approach' (E2) and each should be implemented in a prescriptive manner as required by the TSMS.

MAKE APPROPRIATE TOOL CHOICES [TE.02]:

- **Produce and maintain Coding Standards [TE.02.20]**
  You should ensure that all code developed for the software is written in accordance with a coding standard to ensure consistency and improve the structural quality of the software being produced.

FOLLOW STRUCTURED DESIGN [TE.03]

- **Use only proven components [TE.03.30]**
  When re-using components and libraries, including open source, you should ensure they have a trustworthy software provenance wherever possible. This should include performing a Vulnerability Review (VR) for known vulnerabilities (CVEs) in externally sourced elements both before selection and before deployment.

FOLLOW STRUCTURED IMPLEMENTATION [TE.04]

- **Produce bespoke components in accordance with Coding Standards [TE.04.10]**
  You should ensure that all bespoke software is produced in accordance either with an existing mainstream coding standard or a documented internal coding standard used by your organisation.
- **Use appropriate and recognised data formats [TE.04.20]**
  You should ensure that all data is saved in a format that is independent of proprietary software and follows normal conventions where they exist, for example: ISO 8601 which pertains to the representation and format of dates.
- **Select appropriate algorithms [TE.04.30]**
  You should select algorithms based upon the task that they are meant to achieve in conjunction with the efficacy of the algorithm and the type/quantity of data being handled.

SEEK TRUSTWORTHY REALISATION [TE.05]

- **Review Design for failure modes [TE.05.10]**
  At a minimum (or in addition to design and code reviews) you should conduct a formal review of the design for failure modes, concentrating on high-risk areas. All identified failure modes should be recorded in the R-TSDDL against a unique identifier.
- **Source and configure off the shelf components in accordance with Design/Effect Pattern(s) [TE.05.20]**
  You should use design patterns (a formal description (or strategy) depicting how to solve a problem) across repeatable problems.
- **Ensure mitigations are used for all identified failure modes [TE.05.45]**
  For each failure mode identified as a result of [TE.05.10] above, you should either mitigate against the failure (e.g. by applying a countermeasure) or accept the failure and provide justification for the resultant level of risk.
- **Implement measures to control malicious code [TE.05.65]**
  All externally sourced components, as described in [TE.03.30], should be reviewed for the presence of malicious software (such as viruses, worms, trojans, adware, spyware, scareware) prior to use by a current MalWare scanner that has been publicly reviewed as offering good performance against the WildList.

MINIMISE RISK EXPOSURE [TE.06]

- **Only grant minimum Privileges required, with all other actions defaulting to not permitted [TE.06.10]**
  You should ensure that the software is **only** provided with privileges that are essential to its successful function.
- **Separate program data, executables, and configuration data [TE.06.20]**
  In order to provide defence-in-depth and ensure that the principles of least privilege and need-to-know can be enforced, you should segregate program elements such as program data, executables and configuration files.

PRACTICE HYGIENIC CODING [TE.07]

- **Ensure all variables, pointers and references are properly initialised at first and subsequent uses [TE.07.05]**
  You should ensure that all variables, pointers and references are explicitly initialised before use, that is, they should be assigned an initial value.
- **Ensure all Input Data is Validated [TE.07.10]**
  In order to help protect applications from vulnerabilities (for example: buffer overflows, code injection, directory traversal etc.), you need to ensure that all data input into the system is subject to validation checks.
- **Ensure all Messages are Validated [TE.07.15]**
  It is good practice to use a common set of messages internally within the software, for example internal results and error messages, which are unambiguous and have been validated prior to use.
- **Ensure implementations of all Algorithms are Validated [TE.07.20]**
  It is good practice to use algorithms of known provenance wherever practicable, as this will ensure the effectiveness of the algorithms before use.

- **Ensure all Output Data is Validated [TE.07.25]**
  In order to help protect against accidental leak of information or malicious attacks such as cross-site scripting (where a recipient is sent malicious code from a legitimate site) or SQL injection, all output from your system should be validated before it is sent to a recipient.
- **Ensure Error Handling is implemented comprehensively, and "fails safe and secure" [TE.07.30]**
  To ensure that error handling "fails safe", you need to make certain that all failure modes, exceptions or errors do not leave the software in an insecure state, thereby causing harm to either the system itself or endangering lives.
- **Apply consistent naming convention [TE.07.35]**
  Intelligible naming conventions should be used in order to ensure you produce consistent, clear and well-commented code that is easy to read, understand, and therefore, maintain.
- **Manage resource access explicitly (buffers, stacks, memory, cache and files) [TE.07.40]**
  All access to resources should be explicitly managed to help prevent resource leaks (which could affect performance or cause instability), the unauthorised access of information, inadvertent privilege escalation, improper error handling and insufficient resource tracking.
- **Explicitly remove detritus (temporary files/logs) [TE.07.50]**
  It is important to ensure that the software asset explicitly deletes temporary files on termination of the program, this is not only to enable the re-use of file names and secondary storage, but also to mitigate against vulnerabilities such as TOCTOU and escalation of privileges.

3.3. USE TOOLS EFFECTIVELY (E3)

OBJECTIVES

The objectives of 'Use Tools Effectively' are to use software tools in a consistent and correct manner in order to improve the trustworthiness of the software you use, produce or procure.

FUNDAMENTAL CONTROLS

The following control measures are required to be established (or demonstrated) to meet the TSE objective 'Use Tools Effectively' (E3):

- TSMS (section 2.1)
- TSDDL (section 2.3)

BASIC CONTROLS

The controls listed below meet the TSE objective 'Use Tools Effectively' (E3) and each should be implemented in a prescriptive manner as required by the TSMS.

MAKE APPROPRIATE TOOL CHOICES [TE.02]

- **Selection of appropriate Programming Language(s), considering known vulnerabilities and needs for Typing [TE.02.10]**
  You should choose a programming language that is appropriate for the software you are creating or procuring as this can affect overall development time, ease of maintenance, life-cycle costs and the overall trustworthiness of the finished product. The risks associated with using a particular language should be clearly understood.

- **Selection of appropriate Testing Tools [TE.02.40]**
  You should ensure that the tools chosen have all the necessary functionality for the type of testing you wish to conduct and are suitable for current and proposed environments. Including, for example, a current Susceptibility Review Scanner (SR-S) and any Code Analyser (Binary – BiCA or Source – SoCA).

USE METHODOLOGICAL PRODUCTION [TE.08]

- **ENABLE and use production tool checking features [TE.08.30]**
  Enable production tool checking features: It is important that you enable all of the production tool features, in particular: Error handling; Error Warnings; Exception Handling and Security Checks.

3.4. DEFECT MANAGEMENT (E4)

OBJECTIVES

The objectives of 'Defect Management' are to ensure that the software is free from (or a course of action has been decided for) all identified defects and deviations, thereby ensuring that the software is trustworthy and operates in accordance with all specified requirements.

FUNDAMENTAL CONTROLS

The following control measures are required to be established (or demonstrated) to meet the TSE objective 'Defect Management' (E4):

- TSMS (section 2.1)
- TSDDL (section 2.3)

BASIC CONTROLS

The controls listed below meet the TSE objective 'Defect Management' (E4) and should each be considered and implemented in a prescriptive manner as required by the TSMS.

MAINTAIN DEFECT MANAGEMENT [PR.07]

- **Ensure all Defects identified during Realisation are recorded, reported and assessed, with rectification at earliest opportunity using process for monitoring through a Realisation Trustworthy Software Defect and Deviation List (R-TSDDL) [PR.07.10]**
  All identified defects and deviations should be captured in the R-TSDDL throughout development and subsequently evaluated, prioritised and fixed at the earliest opportunity using a defined process (documented in the TSMS). In particular, during integration testing, this should capture all outputs (together with details of remediation) from the following:
  - Vulnerability Scanner and/or the VR;
  - Malware Scanner;
  - Susceptibility Review Scanner (SR-S);
  - Code Analyser, either Binary (BiCA) or Source (SoCA).
- **Ensure all In Service Defects and Deviations are recorded in an In-service Trustworthy Software Defect and Deviation List (I-TSDDL), reported and assessed, with rectification at earliest opportunity using process for monitoring of deferrals [PR.07.20]**
  All defects and deviations should continue to be captured in the I-TSDDL after the system is 'live', and should be subsequently evaluated, prioritised and patched at the earliest opportunity using a defined process (documented in the TSMS).

ENABLE DEPENDABLE DEPLOYMENT [TE.11]

- **Updating and Patching of implemented software, with routine, critical and emergency options, taking due cognisance of R-TSDDL / I-TSDDL [TE.11.60]**
  In order to help ensure that vulnerabilities in the software are not exploited and that the trustworthiness of your software is maintained, you need to apply patches and updates within an appropriate time-frame, and in accordance with the severity of the patch/update being issued and TS Patching Guidance [AD.05]. Where appropriate, the use of the Common Weakness Scoring System (CWSS) and Common Vulnerability Scoring System (CVSS) can aid in prioritising the remediation of software weaknesses and vulnerabilities.

USE METHODOLOGICAL PRODUCTION [TE.08]

- **Enable and USE production tool checking features [TE.08.30]**
  Use production tool checking features: You should always fix all errors, warnings and exceptions generated by the production tool (whether or not the source code has been successfully translated to object code) prior to live deployment, this is to guarantee that you have made best efforts to ensure that your code is free of defects/deviations.

3.5. ARTEFACT MANAGEMENT (E5)

OBJECTIVES

The objectives of 'Artefact Management' are to ascertain that the required level of trustworthiness has been achieved by the software (pre-release) and to ensure that it is maintained at a comparable level throughout its life-cycle.

FUNDAMENTAL CONTROLS

The following control measures are required to be established (or demonstrated) to meet the TSE objective 'Artefact Management' (E5):

- TSMS (section 2.1)
- TSRA (section 2.2)
- TSRN (section 2.3)
- TSCDM (section 2.3)
- TSDDL (section 2.3)

BASIC CONTROLS

The controls listed below meet the TSE objective 'Artefact Management' (E5) and each should be implemented in a prescriptive manner as required by the TSMS.

UNDERSTAND REQUIREMENTS [PH.02]

- **Implement protection of source code, covering Confidentiality, Integrity and Availability (CIA) [PH.02.10]**
  You should ensure the source code repository is adequately protected from unauthorised access (for example through use of access control), that a version control system is in place with all transactions logged and that where appropriate a software escrow service is used to guarantee availability.

PERFORM TRUSTED SOFTWARE ASSET MANAGEMENT [PR.06]

- **Software assets should be reviewed regularly [PR.06.40]**
  Wherever possible, you should either hold local copies of all software artefacts or make use of an escrow service [TE.11.10]). Furthermore, for each artefact held, regular reviews should be conducted to ensure that access is available, it is adequately protected and the versions held are correct [TE.03.30].

ENABLE DEPENDABLE DEPLOYMENT [TE.11]

- **Ensure source code can be obtained throughout lifecycle, such as by Escrow service [TE.11.10]**
  If you are dependent upon a third-party for the provision of software (for example through commissioned bespoke software), you should consider an escrow agreement in order to ensure the continued accessibility of the necessary artefacts required for the ongoing operation and maintenance of that software throughout its life-cycle.

PERFORM INTERNAL PRE-RELEASE REVIEW [TE.09]

- **Issue formal Trustworthy Software Release Notice (TSRN) [TE.09.50]**
  The TSRN should be produced (and signed by the TSRA) subsequent to Acceptance Testing and should include relevant information from the TSCDM and R-TSDDL together with a summary of the controls considered for each facet of trustworthiness (i.e. Safety, Reliability, Availability, Resilience and Security) during software specification (E1: Scope for Use), development (E2: Coding Approach, E3: Use Tools Effectively and E4: Defect Management) and release (E5: Artefact Management).

4 EVOLUTION

4.1. STATUS/PLAN

This document will be updated on a periodic basis to reflect the constantly evolving nature of the cyber ecosystem and therefore, the need for trustworthiness.

4.2. MAINTENANCE/CONTRIBUTIONS

TSI welcomes input from Stakeholders on all aspects of its activity, including any additions or amendments to the Trustworthy Software Library (TSL).

If you have any suggestions please feel free to engage with TSI, either through your existing contact, or by emailing enquiries@uk-tsi.org.

ANNEX A: REFERENCES

A.1 ASSOCIATED DOCUMENTS

The following TSI documents provide additional information that may be relevant to some or all of the readership.

[AD.1] PAS 754:2014 “Software Trustworthiness - Governance and management - Specification”

[AD.2] TS507-1 “Guide to Risk and TL Assessment”

[AD.3] TS502-1 “TS Essentials Guidance”

[AD.4] “A Guide to Procuring Trustworthy Software”

[AD.5] TS543-1-01 “TS Patching Guidance”

A.2 REFERENCE DOCUMENTS

The following documents from organisations other than TSI provide additional information that may be relevant to some or all of the readership.

[RD.1] “27010:2012 Information technology -- Security techniques -- Information security management for inter-sector and inter-organisational communications”, ISO/IEC, April 2012

[RD.2] “Cyber Essentials Scheme”, HM Government (HMG), April 2014

[RD.3] “15288:2015 -- Systems and software engineering -- System life cycle processes”, International Standards Organisation (ISO) / International Electrotechnical Commission (IEC), May 2015

[RD.4] “12207:2008 -- Systems and software engineering -- Software life cycle processes”, ISO/IEC, August 2008

ANNEX B: ABBREVIATIONS

CIA	Confidentiality, Integrity, Availability
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWSS	Common Weakness Scoring System
FOSS	Free and Open-Source Software
ISMS	Information Security Management System
I-TSDDL	In-service Trustworthy Software Defect and Deviation List
M/E	Mass Market with Explicit Need
M/I	Mass Market with Implicit Need
N/E	Niche with Explicit Need
NVD	National Vulnerability Database
OGL	Open Government Licence
PAS	Publicly Available Specification
PH.NN.NN	Physical (controls)
PR.NN.NN	Procedural (controls)
QAMS	Quality Assurance Management System
QRG	Quick Reference Guide
R-TSDDL	Realisation Trustworthy Software Defect and Deviation List
SA	Self-Assessment
SAQ	Self-Assessment Questionnaire
SAQ-C	Self-Assessment Questionnaire for Components
SAQ-O	Self-Assessment Questionnaire for Organisations
SAQ-P	Self-Assessment Questionnaire for Practitioners
SQL	Structured Query Language
SRS	Software Requirements Specification
TE.NN.NN	Technical (controls)
TL	Trustworthiness Level
TOCTOU	Time-of-Check, Time-of-Use
TSCDM	Trustworthy Software Constraint and Dependency Model
TSDDL	Trustworthy Software Defect and Deviation List
TSE	Trustworthy Software Essentials

TSF	Trustworthy Software Framework
TSI	Trustworthy Software Initiative
TSL	Trustworthy Software Library
TSMS	Trustworthy Software Management System
TSRA	Trustworthy Software Release Authority
TSRN	Trustworthy Software Release Note
VR	Vulnerability Review

ANNEX C: GLOSSARY

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

AVAILABILITY (FACET) The ability of the system to deliver services when requested.

DEFECT A defect can be classed as:

- a coding error/mistake; and/or
- non-fulfilment of an explicit/implicit software requirement.

DEFERRAL A documented and risk managed decision to not resolve a defect or deviation

DEVIATION A deviation can be classed as:

- an unexpected outcome during run-time that is not caused by a coding error or mistake; and/or
- non-conformity with an explicit/implicit software requirement, due to misinterpretation (or otherwise) of the SRS.

PRODUCTION TOOL A program that converts the source code (written in a programming language) into instructions (object code) that the machine understands. As part of this procedure, the following functions may be performed: Pre-processing; Lexical analysis; Syntax analysis/Parsing; Semantic analysis (syntax-directed translation); Code generation; Code optimization; Error Handling; Exception Handling; Security Checks.

RELIABILITY (FACET) The ability of the system to deliver services as specified.

RESILIENCE (FACET) The ability of the system to transform, renew and recover in timely response to events.

SAFETY (FACET) The ability of the system to operate without harmful states.

SECURITY (FACET) The ability of the system to remain protected against accidental or deliberate attacks.

TRAFFIC LIGHT PROTOCOL (TLP) TSI uses the Traffic Light Protocol (TLP), as defined in ISO/IEC 27010:2012 [RD.1], with the following usage convention:

WHITE:	Open Published Material with no restrictions on distribution.
GREEN:	Released Material, not for Website.
AMBER:	Draft (or other relevant) Material, for circulation through direct stakeholder channels and applicable for onward distribution within recipient body.
RED:	Internal (draft or otherwise) material, only for further circulation on reference to originator.