セキュリティ課題規定および
セキュリティ対策方針パッケージ開発ガイダンス文書

第□□版

独立行政法人 情報処理推進機構
注意事項：
・ 本文書は、少ない負荷で、短期間にST作成とその評価ができるようにするために、セキュリティ脅威とその対策方針に関わる記載内容をパッケージ化するためのガイダンス、および参考事例をとりまとめたものであり、参考資料として利用してもうすることを目的に公開するものである。
・ 本文書は、これを参考にして作成されたいかなるSTの評価や認証に影響を与えるものではない。
・ 本文書を参考として利用する限りにおいて、その利用方法を規制するものではない。
GUIDANCE FOR THE
DEVELOPMENT OF
SECURITY PROBLEM
AND SECURITY OBJECTIVE
PACKAGES
FOR COMMON CRITERIA
EVALUATION

Gamma Reference: 4138/3
Date of Issue: 21 February 2005
Author: Dr. M J Nash
Reviewed by: Dr. D F C Brewer

Prepared by
GAMMA SECURE SYSTEMS LIMITED
Diamond House, 149 Frimley Road, Camberley, Surrey GU15 2PS
Tel: 01276 702500 Fax: 01276 692903
Web: http://www.gammassl.co.uk/

© Information-Technology Promotion Agency, 2005
## CC PACKAGES DEVELOPMENT GUIDANCE

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>4</td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Technical Approach</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Baseline</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Companion Documents</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Structure</td>
<td>6</td>
</tr>
<tr>
<td>2. SPD PACKAGES</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Developing Security Problem Definition Packages</td>
<td>7</td>
</tr>
<tr>
<td>3. SCOPING THE PROBLEM</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Sources of Information</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Documenting the Information Obtained</td>
<td>10</td>
</tr>
<tr>
<td>4. THREAT ANALYSIS</td>
<td>12</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>12</td>
</tr>
<tr>
<td>4.2 Deciding on an Analysis Methodology</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Identifying Participants</td>
<td>13</td>
</tr>
<tr>
<td>4.4 Applying the Methodology</td>
<td>16</td>
</tr>
<tr>
<td>4.5 Practical Advice</td>
<td>17</td>
</tr>
<tr>
<td>5. POLICIES</td>
<td>19</td>
</tr>
<tr>
<td>6. ASSUMPTIONS</td>
<td>21</td>
</tr>
<tr>
<td>7. FINALISING SPD PACKAGES</td>
<td>24</td>
</tr>
<tr>
<td>8. SECURITY OBJECTIVES PACKAGES</td>
<td>26</td>
</tr>
<tr>
<td>9. DEFINING SECURITY OBJECTIVES</td>
<td>27</td>
</tr>
<tr>
<td>9.1 Introduction</td>
<td>27</td>
</tr>
<tr>
<td>9.2 Listing All Threats, Policies and Assumptions</td>
<td>27</td>
</tr>
<tr>
<td>9.3 Objectives for the non-IT Operational Environment</td>
<td>28</td>
</tr>
<tr>
<td>9.4 Objectives for the IT Operational Environment</td>
<td>29</td>
</tr>
<tr>
<td>9.5 Objectives Related to TOE Functionality</td>
<td>29</td>
</tr>
<tr>
<td>9.6 Objectives for TOE Development and TOE Assurance</td>
<td>32</td>
</tr>
<tr>
<td>9.7 Producing the Objectives Rationale</td>
<td>33</td>
</tr>
<tr>
<td>10. SELECTING SECURITY REQUIREMENTS</td>
<td>34</td>
</tr>
<tr>
<td>10.1 Introduction</td>
<td>34</td>
</tr>
<tr>
<td>10.2 Selecting Requirements for TOE Objectives</td>
<td>34</td>
</tr>
<tr>
<td>10.3 Selecting Requirements for TOE Development</td>
<td>37</td>
</tr>
<tr>
<td>10.4 Defining Extended Requirements</td>
<td>38</td>
</tr>
<tr>
<td>10.5 Practical Advice</td>
<td>38</td>
</tr>
<tr>
<td>11. FINALISING OBJECTIVE PACKAGES</td>
<td>40</td>
</tr>
<tr>
<td>11.1 Introduction</td>
<td>40</td>
</tr>
<tr>
<td>11.2 Selecting Package Contents</td>
<td>40</td>
</tr>
<tr>
<td>11.3 Producing the Requirements Rationale</td>
<td>40</td>
</tr>
<tr>
<td>12. MANDATED DEVELOPER ACTIONS</td>
<td>42</td>
</tr>
</tbody>
</table>

Version 1.0
Page 2 of 75
CC PACKAGES DEVELOPMENT GUIDANCE

12.1 Introduction............................................................................................................ 42
12.2 ST Introduction........................................................................................................ 42
12.3 PP Claims................................................................................................................ 43
12.4 SPD (Security Environment).................................................................................... 45
12.5 Security Objectives................................................................................................ 48
12.6 Security Requirements............................................................................................ 50
12.7 TOE Summary Specification.................................................................................... 55
12.8 PP Differences......................................................................................................... 58

13 CONCLUSIONS AND COMMENTS.............................................................................. 61
13.1 Introduction............................................................................................................ 61
13.2 Conclusions About the Guidance Offered by this Report.................................... 61
13.3 Other Conclusions.................................................................................................. 61

14 BIBLIOGRAPHY ........................................................................................................ 63
APPENDIX A – EXAMPLE POLICIES............................................................................ 65
APPENDIX B – EXAMPLE ASSUMPTIONS.................................................................... 68
APPENDIX C – EXAMPLE OBJECTIVES....................................................................... 72
1 INTRODUCTION

1.1 Purpose

This Report provides guidance to help develop Protection Profiles (PPs) and Security Targets (STs), specification documents used to prepare for evaluation under the Common Criteria (CC) Evaluation Scheme, or for evaluation against the corresponding International Standard, ISO/IEC 15408.

This guidance is illustrated by a worked example, which will be found in Gamma Report reference 4183/4, Example of the Development of Security Problem and Security Objective Packages for Common Criteria Evaluation [1].

1.2 Technical Approach

This report is intended to provide a methodological approach to the development of PPs and STs. The approach is based on two concepts, phasing and packaging.

Using this methodology, the development of PPs and STs is broken down into three phases, to be performed sequentially:

- Specification of the security problem addressed by the PP or ST as a Security Problem Definition package (SPD);
- Specification of the security solution to the problem as an Objectives Package (OP);
- Implementation details (found only in STs).

This Report does not address implementation details, such as the production of the TOE Summary Specification. However, one chapter of the Report covers in outline all developer actions required to prepare a PP and ST, in order to put the use of SPDs and OPs in context, and this does address the TOE Summary Specification.

For both SPDs and OPs, specification work is subdivided into a series of tasks. This simplifies correspondence tracking and permits analysis work to be verified as early as possible, before proceeding to the next task. Both SPD and OP packages are self-contained and can therefore be tested for completeness and internal consistency.

Other approaches to developing PPs and STs exist. Much of the detailed guidance within this document will be applicable to other methodologies.

This Report assumes a knowledge of basic CC concepts, as found in Sections 4.3 and 4.4 of Part 1 of the Common Criteria Version 2.2 [2].

1.3 Baseline

This Guide uses Version 2.2 of the Common Criteria as its baseline. This was the latest version officially published at the time of preparation of the Guide. This version is
identical to Version 2.1 [3], the most widely distributed and used version, but including error corrections and interpretations up to December 2003.

The current version of the International Standard, ISO/IEC 15408:1999 [4], is technically identical to CC Version 2.1, but with some presentational changes to follow standards development practices.

A new version of the International Standard corresponding to the Common Criteria, ISO/IEC 15408:2005 [5] was undergoing ballooning at the time of publication of this Report. It is intended that this revised International Standard will be technically identical to CC Version 2.2, but will include further agreed interpretations up to December 2004. It will be fully reformatted into the presentational style normally used for International Standards.

It is expected that a revised version of the Common Criteria, CC Version 3.0, will be published in 2005 or 2006, initially for trial use. An unofficial partial revision, Version 2.4 [6], is available for examination. This contains a complete revision of the PP and ST specifications of Versions 2.1 and 2.2, based on ideas originally published in the proposed ASE supplement to the Common Criteria. Where possible, this Guide takes account of this work. Several of the methodological improvements of Version 2.4 are adopted within this Guide; in particular, the package approach.

One of the objectives of this Guide is to show that these improvements can be used in practice. It is hoped that this Guide will be readily adaptable for use with CC Version 3.0.

1.4 Companion Documents

This document is accompanied by a worked example that illustrates the proposed methodology. It will be found in Gamma Report reference 4183/4.

There is an ISO Technical Report, ISO/IEC TR 15446:2004 [7], usually referred to as the PPST Guide, that contains guidance on the production of PPs and STs. This document is based on ISO/IEC 15408:1999, equivalent to CC Version 2.1, and was prepared before much practical experience of PP and ST development was available. However, it contains much useful guidance and is an essential companion to this Report. This Report indicates clearly all places where its guidance significantly disagrees with or contradicts the PPST Guide.

The Security Problem Definition package concept used in this Report was first expressed in the ASE Security Target Evaluation supplement to the Common Criteria [8], published in May 2002. This proposed amendment to the CC was not adopted, and later versions of the CC have incorporated only part of its approach.

The US Government’s National Information Assurance Partnership (NIAP) has published two Consistency Instruction Manuals ([9],[10]) to identify frameworks of consistent security requirements for use when developing PPs for US Government use. Although these documents are specifically aligned to US Government needs, they do contain some relevant advice to all PP and ST developers.
1.5 Structure

There are fourteen chapters and three appendices in this Report, including this introduction. The next six chapters address the creation of Security Problem Definition packages. The following four cover the creation of Objectives Packages. The next chapter provides outline guidance for all developer activities required in preparing PPs and STs. The last major chapter presents the conclusions resulting from the application of the methodology and the preparation of this Report. The final chapter contains a bibliography. The three appendices contain examples of policies, assumptions and objectives.
CC PACKAGES DEVELOPMENT GUIDANCE

2 SPD PACKAGES

2.1 Introduction

This following five chapters of this Report deal with the first phase of preparing a PP or ST, namely identifying and specifying the security problem or problems that need to be addressed. From the point of view of CC evaluation, the security problem is axiomatic, i.e. the process of deriving the security problem definition falls outside the scope of the CC.

However, this phase is the most important in preparing for evaluation. To quote from the Trial Use Version 2.4 of the CC:

“The usefulness of the results of an evaluation strongly depends on the ST, and the usefulness of the ST strongly depends on the quality of the security problem definition. It is therefore often worthwhile to spend significant resources and use well-defined processes and analyses to derive a good security problem definition” ([6], Part 1, A.6.1).

If the problem specified is the wrong problem, or if it is ambiguously described, then the remainder of the PP or ST will also be wrong. Worse, the wrong system or product may be selected or purchased on the basis of a valid but incorrect specification.

2.2 Developing Security Problem Definition Packages

The Common Criteria does not mandate or recommend any particular methodology for developing PPs and STs once a security problem has been identified.

This Report proposes a specific approach, based on the concept of packages. A package is a self-contained piece of work that can be developed in isolation from the rest of the PP or ST, and can potentially be reused if other aspects of the PP or ST change.

This methodology proposes that the first type of package to be developed should be a complete specification of each security problem to be addressed by the PP or ST, the Security Problem Definition (SPD). The benefit of this approach is that the problem can be validated and agreed before consideration of other issues.

A PP or ST will normally address only a single security problem; however, separation into multiple SPD packages may be appropriate in some circumstances; for example, where there are multiple sponsoring organisations who are only interested in their own requirements, or where a composite system is physically distributed and different component operate in different environments with differing requirements.

Using the methodology of this Report, the process of defining an SPD package is made up of five steps, performed in order:

- Scoping the problem to be addressed by the SPD;
- Performing a threat analysis;
- Defining applicable policies;
- Defining applicable assumptions;
• Finalising the SPD.

Each of these steps will be described in following chapters. The PPST Guide suggests activities are performed in a different order – however, the order above seems to work best in practice.

Within CC Version 2.2 there is no concept of a security problem definition package; the necessary information is held within several different sections of the PP or ST. Despite this, using the SPD concept will simplify and assist in PP and ST production for evaluation against CC Version 2.2. Guidance in Chapter 12 explains how to distribute the relevant information within the officially recommended PP and ST structure.

Trial Version 2.4 introduces the concept of a Security Problem Definition section within PPs and STs. However, its proposals are a weakened version of the SPD concept originally found in the ASE Supplement, which treated the SPD as a package that could be considered in isolation. The SPD packaging proposed in this Report is a development and expansion of the original ASE approach.
3 SCOPING THE PROBLEM

3.1 Introduction

The first step in specifying an SPD package is to identify and describe the security problem it addresses in general terms. There are always many things about a security problem – and its intended solution – that are already fixed and known before SPD definition begins.

3.2 Sources of Information

3.2.1 Introduction

There are many ways that aspects of the security problem can be identified. The subsections following discuss some of them. In a particular organisation, there may be others that a generic methodology cannot identify.

3.2.2 Required Functionality

Security functionality may be part of the purpose of the system or product under consideration. This particularly applies to products, where security services to be available to the purchaser through Application Program Interfaces (APIs) or Human Computer Interfaces (HCIs) may be an essential part of the product specification.

If security functionality is part of the user requirement, providing it is part of the problem addressed in the SPD.

3.2.3 Risk Assessment

A security risk assessment may have already been performed covering a proposed system, and even a product, and identified risks that need to be reduced by IT security controls. These risks represent part of the security problem.

There are many methodologies for performing risk assessments. However, these methodologies generally accept that for a risk to exist, there must be three things: an asset with a value that can be damaged in some way, a threat, something or someone who can damage the asset, and a vulnerability, a way that the asset can be damaged. If any one of these three does not exist, there can be no risk. This model is assumed by the CC; if the actual risk assessment did use an incompatible model of risk, there might be problems mapping its results into a suitable form for use in the SPD.

3.2.4 Threat Assessment

A threat assessment is a weakened form of risk assessment where it is assumed that if a threat exists, assets can be damaged and thus a risk will exist. In this case, the threats represent the security problem.
Threat assessment is particularly appropriate where the person trying to identify and specify a security problem is not the owner of the assets that will be protected, and thus not in a position to perform risk assessment or determine the value of assets.

3.2.5 Management Policy

A security problem can result from a policy decision by management, for example that all systems in a particular organisation will contain certain standard IT security controls. This process is sometimes known as “minimum standards” or “risk avoidance”. The policy may be arbitrary, for example, following what similar organisations do, or it may have a logical basis, for example to meet legal requirements or contractual conditions imposed by customers.

Of course, even where a policy has a logical basis in law or contract, the mandated security controls may not be appropriate for a particular system or organisation, or may only be applicable in part.

3.2.6 Presentational Policy

A security problem may arise from a wish to demonstrate that an organisation or a product implements certain IT security controls. This policy may arise due to marketing needs, or from a wish to be seen to follow best practice.

Security problems of this type are well suited to CC evaluation, as successful evaluation using an approved evaluation facility will permit an official certificate to be issued, providing independent proof that the controls exist. Published PPs can be used to identify suitable controls.

The drawback to policies of this type is that they are based on achieving certification or demonstration of compliance, not in selecting security controls that are relevant to the system or product concerned. This can cause problems finding reasons to put in the SPD that justify the need for the controls. They may have to be treated as policy decisions, which the originator may be reluctant to acknowledge is the true reason for their selection.

3.2.7 Evaluation Policy

An organisation may have a policy that systems or products are evaluated using the CC Scheme, regardless of the IT security controls they implement.

This requirement is problematic. The security problem to be addressed forms no part of the policy and is therefore not properly defined. However, such policies are found in practice, and do result in requirements for STs to be prepared.

3.3 Documenting the Information Obtained

The best source of information about a security problem is the results from a security risk assessment. Not only is this likely to be comprehensive, but most risk assessment methodologies introduce the concept of proportionality, where risks can be tolerated, so
long as the likelihood of a loss is very low or the consequences of a loss are not significant. Identifying both acceptable and unacceptable risks enables the security problem to be modified later through design trades. If the controls required to eliminate particular risks turn out to be difficult to implement or difficult to evaluate, an acceptable overall level of risk can still be achieved by using different controls in different ways to counter different potential risks.

If describing part of the problem in terms of risks is not possible, it is almost certain to have an arbitrary basis that cannot be modified or amended. It is important that this is made clear in the description.

Relevant information may relate not only to the system or product to be developed, but also to its operating environment. The operating environment determines the level of reliance that can be placed on personnel, procedural and physical controls. A public space is very different in its security needs to a locked server room. If it has been established that certain personnel, procedural and physical controls can be assumed to be in place, that will be important part of scoping the security problem.

As well as information about risks and controls, design decisions may have already been made about how certain security functions are to be implemented – for example, a decision to use biometric authentication rather than passwords, or to use certain communications protocols such as http/https that have defined security characteristics.

Some parts of a security problem may not be solvable by technical means; they may only be countered by personnel, procedural and physical controls. They are still part of the security problem, and need to be documented. Indeed, any aspect of the security problem that has already been decided should be documented.

When all the information available has been identified, collated and checked for inconsistencies, it will be found to fall into two categories:

- things that the product or system must do; and
- things that the product or system need not do.

This distinction is important. Within the SPD, things that the product or system must do become Organisational Security Policies (OSPs). Things that the product or system need not do become Assumptions. These two types of constraints are discussed in detail in later chapters. Expressing these requirements in suitable terms for inclusion in the SPD is covered in these later chapters. But if the requirements are fixed and certain, they can be specified and formatted as suggested in those chapters at this stage. This may help in threat analysis, the next step.
4 THREAT ANALYSIS

4.1 Introduction

Once the security problem has been scoped and documented, the next step is to perform a threat analysis. The CC does not prescribe any particular methodology for identifying applicable threats. However, the analysis must identify all the threats perceived as relevant to the TOE in question.

Using the approach of this Report, threat analysis is particularly important, as most Policies and Assumptions will be justified in terms of threats eliminated from further consideration. This will be explained more fully later in this chapter.

In order to perform a threat analysis, it is necessary to perform three activities:

- decide on the analysis methodology to be used;
- identify the participants required by that methodology;
- apply the methodology.

These activities are discussed in turn in subsequent sections of this chapter.

4.2 Deciding on an Analysis Methodology

The best methodology to identify the applicable threats will depend on how the security problem was defined. If the problem was specified in terms of the results of a risk assessment, then a list of threats may already be available. Even if this is not the case, it may still be possible to identify the threats from other existing and available information.

Unfortunately, in most cases sufficient information will not be available, and a formal threat analysis must be performed.

This Report recommends an approach that was developed for the Japan Information Technology Security Evaluation and Certification Scheme (JISEC). This approach is based upon a structured database of generic threats, together with a technique called threat tree analysis (see [11]). To save duplication of information, this recommended approach is not described in detail within this Report. The information is available from JISEC.

Threat tree analysis is not always appropriate and a number of other threat identification techniques are identified in [11]. The best technique to apply to a particular security problem will depend on the nature of the security problem.

If the security problem and its surrounding environment are both well defined, constructing a threat tree is usually the most effective approach. Where the problem is defined in general terms, or the environment is uncertain or arbitrary, a simple serial search of the generic threats database may suggest applicable threats more efficiently than a top-down analysis. This particularly applies to product developers, who do not
have knowledge of the actual environments in which their products will be used. Full
details of these and other approaches are given in [11].

4.3 Identifying Participants

4.3.1 Introduction

Although CC Version 2.2 only requires that each threat is identified and explained ([2],
ASE_ENV.1.2C and APE_ENV.1.2C), CC Trial Version 2.4 ([6], ASE_SPD.1.2C and
APE_SPD.1.2C) requires that each threat is characterised in terms of a threat agent, an
asset and an adverse action. This form of description should be used where possible –
with the interpretation that asset is understood to include types of asset, since the actual
assets may often not be defined. CC Version 2.2 and the PPST Guide suggest that
threats should also be characterised in terms of attacks ([2], Part 1, Section 4.3.1), but in
practice attack methods are not obvious until the security controls have been selected,
and so this information is not available for use in threat analysis.

4.3.2 Threat Agents

The JISEC Database is organised on the basis of the following types of threat agent:

- Attackers;
- Authorised Users;
- Privileged Users;
- Administrators;
- System Owners and Developers.

These are defined in [11], and form a suitable set for use in threat analysis.

The list of threat agents above does exclude one possible type of threat agent that may
be relevant to some security problems – acts of nature (sometimes called “acts of god”),
such as earthquakes, where there is no human agent involved. [11] treats such threats as
being the responsibility of the system owner and developer, although they are not
involved in formulating or executing any attack. In some cases, describing the related
agent as “none” or “nature” may be clearer or more acceptable to the system owner.

In particular security problems, different participants within the groups identified above
may have distinct characteristics that mean that they have to be separated out. This
should be obvious from the description of the security problem.

4.3.3 Types of Asset

Assets are important to threat analysis and need to be properly identified. Most threat
analysis methodologies can handle imprecision or overlap in players and adverse
actions, but assets need to be distinct and well described. The JISEC Database does not
try to identify generic assets. In consequence, this section therefore offers a detailed
methodology to identify the assets or types of asset that need to be protected in a
particular TOE.
In the case of a system, it will often be possible to identify the precise assets to be protected, as this will form part of the definition of the system. In the case of a product, the actual use of the product is often not known, and it is therefore only possible to identify the types of asset that the product is intended to protect.

Assets associated with IT systems usually fall into one of three classes:

- information;
- processes;
- physical.

Information assets represent data that is of value to the owning organisation. Examples of possible types of information assets are:

- General User Data;
- System Data;
- Specialist Databases;
- Client Data.

Specialist databases would represent information that is only of value to some users. Examples might be a personnel database (only of value to the human resources department) or a customer database (only of value to the order processing and marketing departments). Client Data might represent data not owned by the owner of the system and for which there is a special and relevant characteristic, a legal duty of care.

In the case of a system, it will normally be possible to identify the names and characteristics of the actual databases or other information assets to be protected.

In the simplest case, all data can be treated as being of equal value and at equal risk of attack, and represented by a single information asset, named something like “user data”.

It is often necessary to distinguish system data, i.e. data used by the technical security functions (TSF) of the TOE, from other data. If system data is modified or deleted, the TSF functions may operate incorrectly, and permit other types of attack, whereas if other data is modified, only the data directly involved is corrupted, the TSF continue to function, and will continue to protect other assets.

It is quite common for these two information assets to be sufficient, one representing TSF data and the other all other data protected by the system or product.

Sometimes different types of TSF data may be susceptible to different attacks, or have different consequences if compromised, and thus required to be distinguished. Examples of distinct types of system data might be:

- TSF configuration data;
- The authentication data database;
- Audit records.
Sometimes very limited and specific forms of data that are susceptible to specialised attack may need to be distinguished; for example, cryptographic keys.

Process assets represent applications, where data is transformed or analysed. The distinction from information assets is that the associated data is of little value without the processing capabilities of the related applications. Examples of possible types of process assets are:

- Financial;
- Communication;
- Logistical;
- Manufacturing;
- Office Automation.

Financial applications might include payroll, investment management or accounts management. Communication systems might include e-mail or intranet/extranet information handling. Logistical systems might include order processing, warehouse control or resource scheduling. Manufacturing applications might include real-time process control. Office Automation might cover structured text processing.

In the case of a system, it will normally be possible to identify the names and characteristics of the actual processes to be protected.

In general, process assets are only susceptible to modification or denial of service attacks. For example, the functionality of the associated applications software could be altered, perhaps to remove authorisation checks or to alter financial processing. A single asset, called “applications software” or something similar, is usually sufficient to cover all processes.

Physical assets represent the actual information processing equipment used to support the information and process assets. Examples of possible types of physical assets are:

- Critical Network Infrastructure;
- Portable PCs;
- Data Centres.

It is very unusual for TOEs to offer protection of physical assets as part of the security problem. In consequence, it is therefore unusual for physical assets to appear in PPs or STs. However, there are applicable techniques, such as automatic closedown on power failure, that could offer protection to physical assets and in such cases physical assets might appear in the PP or ST.

It is important not to identify too many assets or types of assets. If two assets or types of assets have the same potential for attack and consequences of attack, they should be grouped together into a composite asset type. Many TOEs will protect only two types of asset, TSF data and user data. More than six types of asset is probably inappropriate for anything other than a TOE that is expected to offer very complex or individualised protection capabilities.
As part of the definition of the security problem, certain assets or types of assets may have been excluded from requiring protection. If this is the case, they should be listed separately: this information will be needed later to explain why they have been excluded from the threat analysis.

4.3.4 Adverse Actions

The JISEC Database identifies the following types of adverse action:

- Improper access;
- Improper transmission of access rights;
- Denial of legitimate access;
- Non-accountability.

These are defined in [11], and form a suitable set for use in threat analysis. CC Version 2.4 suggests that adverse actions are described in terms of the consequences of a successful attack, e.g. loss of confidentiality ([6], Part 1, Section 4.1.1), but this may be unnecessarily specific and limiting.

In particular security problems, particular adverse actions within the groups identified above may have distinct consequences that mean that they have to be separated out. This should be obvious from the description of the security problem. There may also be other, specialised, types of adverse action that do not fall naturally into the groups above. This should also be obvious from the description of the security problem and will again need to be treated separately.

4.4 Applying the Methodology

Once a methodology has been selected, and the necessary information to apply that methodology has been prepared, the next step is to apply it to generate a list of applicable threats.

In practice, many possible threats can quickly be discounted. There are two particular techniques that are very useful – identifying excluded or tolerated threats, and identifying threats already covered by policy.

Many types of threat will have already been discounted as part of the definition of the security problem, either because they have been excluded from the scope of the TOE, or a decision has already been made to tolerate them because the impact of associated risks is low, or has been transferred to a third party (e.g. an insurer).

Exclusion is common in the context of products – for example, the vendor of an operating system may just decide not to include anti-virus (AV) protection within the product, assuming that the purchaser will wish to buy a supplemental specialist AV product, or will use the product in an environment that is isolated from infection.

Tolerating threats is usually found in the systems context; it requires assets to be valued, something a product manufacturer cannot do.
The relevant information to discount threats is usually obvious from the list of things that the system or product need not do. If not, it needs to be confirmed and then added to that list. It should also be recorded in the form of an Assumption (see Chapter 6).

In many systems or products, a decision will have already been made to include security functionality, independent of the analysis of actual threats. It is common in the case of products – for example, an operating system vendor will normally include user identification and authentication functions, even if the product is designed for single user situations.

If this mandated functionality will counter a particular type of threat, that threat need not be investigated further to see if it is actually applicable; protection will be provided regardless.

The relevant information to ignore threats is usually obvious from the list of things that the system or product must do. If not, it needs to be confirmed and then added to that list. It should also be recorded in the form of a Policy statement (see Chapter 5).

All remaining threats must be identified and considered, and a full list of applicable threats produced, describing each threat in terms of agents, assets and adverse action.

Some threats may be applicable to a particular TOE, but it has already been decided as part of scoping the security problem that they will be countered by security controls within the operational environment. It may only be possible to counter some threats by measures in the environment (for example, where physical protection is necessary). These threats still need to be listed, but it is worth making a note with the entry that they will generate environmental objectives; this information will be very useful later.

However do not prejudge how threats will be countered if it could be done by either the TOE or its environment. This would take away the ability to make design trades later when controls are being selected and designed.

Threats to the development of the TOE and to its development environment generate requirements for assurance objectives. In CC evaluation, these threats are usually handled by a policy decision to use a particular Evaluation Assurance Level (EAL), without detailed consideration of the actual threats. However, the JISEC threat analysis methodology is equally applicable to the development environment, and the JISEC Database contains details of relevant threats.

### 4.5 Practical Advice

Threats indicate possible ways that the TOE might be attacked. Therefore they should be worded as such. The best way to do this is to use a verb form such as “may”. For example:

*T.NOAUTH*  An unauthorised person may attempt to bypass the authentication controls of the TOE so as to access and use TOE resources.*
It helps to start each threat description with a name for reference purposes. By convention, most PP and ST authors start threat names with “T.” to assist identification. Descriptions should be kept short and to the point. The database entry for each threat in the JISEC Database contains a suitable name and short description.

Methodologies must not be used blindly. They must be adapted and interpreted to meet the requirements of a particular security problem. Do not be afraid to go back and start again using a different approach if a particular form of categorisation is not working out in practice.

Threats can be combined if their agents, assets and adverse actions are similar. This will reduce the size of the threat list and save time later, since the same controls will often be used to counter them. Equally, where a threat has markedly different impacts depending on factors like threat agent or asset involved, it will clearer and save time later if the threat is split into multiple threats that are more specifically worded.

Information indicating that threats can be discounted is often expressed indirectly. For example, consider the statement:

*Administrators can be assumed to be non-malicious, trustworthy and competent.*

This is expressed in terms of a threat agent, and effectively discounts most types of threats normally associated with that type of agent. Some of these types of threats are specific to administrators and can therefore be discounted. Other types of threats will still apply, but can be restricted to other applicable threat agents only e.g. ordinary users. Do not forget to add the assumption that reduced the scope of these threats to the list of assumptions.

In some cases it may not be possible to identify threat agents or adverse actions – only that the associated risk is unacceptable. In these cases, it is pointless to create characterisations based on guesswork or imagination. The threat is unacceptable by definition of the security problem, and should be identified and justified as such.

Once a final list of threats has then been prepared, it should always be checked for completeness and consistency. If a threat has been broken down by type of asset or type of threat agent, are all possibilities covered? Are similar threats treated in a similar manner? If not, is there a good reason? Although inconsistencies and omissions may well be detected later in the preparation of the PP or ST, checks now will save time and reworking later.

It is possible that threat analysis may identify no threats to be listed as applicable to the TOE. This can happen, for example, in PP’s that are designed to meet general corporate or government policies and nothing else. This is perfectly acceptable in CC evaluation; the threats section should be left blank, with an indication that no specific threats were identified.
5 POLICIES

Policies are statements of things that the system or product will do, regardless of consideration of threats or other matters. Therefore they should be worded as such. Standards written in English use the verb form “shall” to indicate requirements of this type. Most English speakers find this unnatural, and the verb form “will” is perhaps to be preferred. Thus an example of a clear and well worded policy might be:

P.IDAUTH Administrators will authenticate themselves before accessing any TOE functions or data.

As for threats, it helps to start each policy with a name for reference purposes. Descriptions of policies should be kept short and to the point. By convention, most PP and ST authors start policy names with “P.” to assist identification.

In the CC, policies are often referred to as Organisational Security Policies, or OSPs for short. This wording is not recommended – some policies may only apply to a particular system, rather than all systems within the organisation.

Most applicable policies will have been identified during scoping of the security problem, or during threat analysis. However, a full investigation should be made to identify any other policies that are relevant to the security problem.

Policies are used to specify:

- Mandatory security functions to be incorporated within the TOE;
- Mandatory technologies/techniques to be used to implement particular security functions (which implicitly requires those functions to be present).

Policies can also be used to replace threats. This is appropriate if:

- It is not certain that a particular threat exists, but a policy decision has been made to protect against it regardless;
- A policy decision has been made as to how a particular threat will be countered, e.g. by specifying:
  - what controls will prevent a successful attack;
  - what will be done if an attack occurs;
- A policy decision has been made to adopt a particular approach to countering a number of related threats.

The PPST Guide does not cover all of the above uses for policies.

There is no value in replacing a threat with a policy unless there is some additional information represented in the policy that is not implicit in the statement of the threat.

New policies identified at this stage may require changes or reworking of previous activities, e.g. to identify and delete threats that are now covered by policies.
In practice, most policies are easy to identify and express clearly. Examples of policy statements, both good and bad, will be found in Appendix A of this Report. However, there are some common problems that should be noted.

Policy statements are sometimes misused to express requirements for things that the TOE must not or cannot do, but which instead must be enforced by the operational environment of the TOE. If a requirement cannot be implemented by the TOE, the correct way to specify it is as an Assumption on the operational environment (see chapter 6). If a proposed policy cannot be enforced by either the TOE or the operational environment, then the policy statement is either meaningless or unachievable.

During the course of specifying the security problem and its solution, the boundary of the proposed TOE may need to change, to transfer functions from the TOE to its operational environment or vice versa. This may cause policies to become assumptions or assumptions to become policies, or it may require policies or assumptions to be re-specified to take account of the new TOE boundary. Similarly, in composite TOEs that are broken down into several components addressing different security problems, an assumption for one component is often implemented by another as a policy requirement. In such cases, careful wording of the policy statements will enable them to be reused in the other SPDs as assumptions, ensuring compatibility and easy consistency checking.

Sometimes it is not clear during SPD preparation whether a policy will be implemented by the TOE or by its operational environment. This is acceptable; it can be resolved during OP definition when the requirements for security functionality are finalised. Both TOE objectives and environmental objectives can link back to policies. A policy may even be partially implemented by the TOE and partially by the environment.

Not all security problems require policies. This is perfectly acceptable in CC evaluation; the policies section should be left blank, with an indication that no applicable policies were identified.
6 ASSUMPTIONS

Assumptions are statements of things that the system or product need not do, regardless of consideration of threats or other matters. They should therefore be worded as statements of fact. An example of a clear and well worded assumption might be:

A. PHYSICAL. The TOE is located in a physically secure location.

Assumptions have two uses:

- To indicate that a particular control or type of control will be provided by the operational or development environment, and not the TOE;
- To indicate that particular threats or type of threats can be discounted.

The first of the above types is best expressed using the verb “will”, as it implies a control must be provided. The second form is best expressed using an active, present tense verb such as “is”.

Every assumption should be given a short name for reference purposes. Descriptions of assumptions should be kept short and to the point. By convention, names of assumptions start with “A.” to assist identification.

The PPST Guide does not recognise the use of assumptions to discount threats.

In practice, it is more difficult to express assumptions clearly and positively than it is policies or threats. Avoid the temptation to use verbs such as “may” or “should”: assumptions are statements of fact. Keep assumptions about the environment distinct from assumptions about discounted threats. Some examples of assumptions, both good and bad, will be found in Appendix B of this Report.

Assumptions about the operational environment need to be separated into the three areas of:

- physical protection;
- personnel and procedures;
- technical functionality outside the TOE.

CC Version 2.2 refers to physical, personnel and connectivity aspects of the environment, as it considers that only connectivity issues require assumptions about technical functionality external to the TOE. However, practical experience has shown that this is inadequate. Many external technical controls do fall under the heading of connectivity, for example:

A. INTERNET. The TOE will be isolated from the Internet.

However, other assumptions about technical controls are often necessary. For example:

A. NO_DEV_TOOLS. No tools will be present in the operational environment of the TOE that permit ordinary users to add new functionality to the system.
Note that the U.S. Consistency Instruction Manuals [9] and [10] are even more restrictive, limiting assumptions to non-technical measures only – although then giving an example similar to A.NO_DEV_TOOLS above which is clearly IT related.

In many cases policies and threats will be partially handled by the TOE and partially by the environment. For example, technical controls within the TOE may need supporting procedural or physical measures to be present in order to work effectively. The need for such supporting measures in the environment must be identified and expressed as assumptions.

The U.S. Consistency Instruction Manuals suggest that supporting requirements should be specified as policy statements (e.g. “The TOE security environment will contain a secure time source.”), rather than assumptions, although this contradicts their own guidance in the same instruction that all policies must link, at least in part, to a TOE objective. This Report strongly recommends that such controls are expressed as assumptions.

Assumptions that identify threats that are going to be discounted or ignored are not used during evaluation. However, they are valuable in showing consistency and completeness. Where threats may have been identified by a methodological approach, these assumptions are need to show complete coverage in the rationale. A threat may be partially discounted and partially countered. In this case the assumption is needed when tracing back the security objectives for the countered part back to the threat to show that complete coverage is provided.

Many assumptions will have been identified during scoping of the security problem, or during threat analysis. However, a full investigation should be made at this stage to identify any other assumptions that are relevant to the security problem. When a decision is made that a policy will be implemented, or a threat countered, by the environment, this must always be recorded as an assumption. These assumptions should be worded to reflect the policies and threats in question, as they will generate objectives for the environment that will need to match those policies and threats.

One assumption can often be used to counter multiple threats that are related in some way. If a threat tree approach has been used, where multiple detailed threats all to be countered by the environment share a common hierarchical node further up the tree, the assumption can be expressed at the level of the shared node. For example, if all threats resulting from adverse actions by administrators are discounted, this can be expressed in one single assumption:

\[ A.NO.POOR.ADMINISTRATION \quad \text{Administrators have the necessary skills, training, time and resources to perform all their allocated administrative functions, and perform all those functions correctly.} \]

When formulating assumptions, a good test for a well formed and necessary assumption is that if the statement is untrue, the TOE could be successfully attacked.

Separating assumptions by type will be helpful when identifying and specifying objectives. Assumptions about the development process and development environment should be separated out first. The next category should cover personnel, procedural and
physical assumptions related to the non-IT operational environment. This should be followed by the technical functionality assumptions about the IT operational environment. Finally, the assumptions about discounted threats. These should be kept fully separate as these do not generate objectives at all.

Not all security problems require assumptions. This is perfectly acceptable in CC evaluation; the assumptions section should be left blank, with an indication that no required assumptions were identified.
7 FINALISING SPD PACKAGES

The last stage of SPD production is finalising the SPD packages. This involves two tasks:

- Preparing a complete list of all threats, policies and assumptions, and, if multiple SPD packages are required, assigning them to particular SPD packages.
- Performing consistency and completeness checks to confirm the packages accurately represent the security problem or problems addressed.

Each SPD should consist of:

- An introduction to the security problem addressed by the SPD, that can serve as the ST or PP overview.
- A list of all applicable threats, policies and assumptions, together with any necessary explanations as application notes.
- A rationale based on the completeness and consistency checks demonstrating completeness and coverage.

There is no requirement in the CC to provide an SPD rationale; the statement of threats, policies and assumptions expressed in the SPD is treated as axiomatically correct for the purposes of evaluation. However, it is strongly recommended that a formal rationale is produced; if requirements change, or complications are found later on, the rationale will make SPD reworking much simpler and reduce the risk of introducing errors.

Similarly, there is no requirement in the CC to identify threats that have been discounted or ignored. Once again, this information is extremely useful if circumstances change and the SPD has to be reworked. This Report recommends that appropriate assumptions about such threats are always included.

Multiple SPD packages are only useful where a single TOE contains multiple independent components that operate in more than one distinct operational environment; for example, where complex systems cover multiple and different physical locations. In these cases, separate SPDs for each component will assist in independent development of each component and in providing consistency checking. Assumptions about technical functionality in the environment of one component should correspond to policies implemented by other components. Where multiple SPDs are prepared, each one should be made fully self-contained and not require knowledge of the internal contents of others.

Consistency and completeness checking involves checking that all constraints and requirements found whilst scoping the security problem have been reflected in policies or assumptions, and that all identified threats have been countered or discounted in some way. Similarly, all policies, threats and assumptions listed in the SPD should be traced back to requirements of the original problem. Creating cross-reference tables is often an efficient and easy way to show that consistency and completeness exist.

If multiple SPD packages are created, they must be checked to confirm all of the threats, policies and assumptions of the original security problem are addressed, and that the
treatment of those threats, policies and assumptions is consistent across all the SPD packages created.

Assumptions and policies may appear to conflict, i.e. a firm policy requirement “will do X” may appear to be contradicted by an assumption “need not do X”. On inspection, it will usually be found that there is no actual conflict, just greater explanation and precision of wording is required. If there is a direct conflict, it must be resolved by re-examining the security problem.

Where a policy covers or replaces a threat or threats, it is then pointless to include those threats as part of the list of threats within the SPD. However, the threat must still be addressed in the source rationale, to show that it has been considered and is covered. The correspondence between these threats and the stated policies needs to be included in the rationale, in case policies are revised in the future.
8 SECURITY OBJECTIVES PACKAGES

The next three chapters of this Report deal with the next stage in PP and ST production, defining Objectives Packages (OPs). In CC terms, OPs cover two aspects of ST or PP specification, namely security objectives and security requirements, including the definition of any extended requirements components that may be needed.

The process of defining OPs is made up of three steps:

- Defining the security objectives corresponding to an SPD;
- Generating Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) to meet the objectives;
- Assigning the objectives and related requirements to packages.

Each of these is covered by one of the following chapters.
9 DEFINING SECURITY OBJECTIVES

9.1 Introduction

Security objectives should consist of short, clear statements that together define a high-level solution to the security problem identified in the related SPD. They should be worded as requirements. In English, the verb form “must” is a good way to word objectives. Examples of security objectives, both well and badly worded, will be found in Appendix C.

The tasks required to define security objectives are:

- Prepare a list of all threats, policies and assumptions to be covered by the objectives;
- Identify the objectives for the non-IT operational environment;
- Identify the objectives for the IT operational environment;
- Identify the functional objectives for the TOE;
- Identify the assurance objectives for the TOE and its development environment;
- Produce an objectives rationale linking back to the identified threats, policies and assumptions.

These will be described in following subsections. They are usually best performed in the order given above, although the PPST Guide prefers a different order.

Objectives are often interlinked, sometimes in ways that are not immediately obvious. It is therefore best to leave assigning objectives to OPs until after the security requirements for TOE objectives have been selected. These often highlight dependencies and interrelationships.

9.2 Listing All Threats, Policies and Assumptions

The first task is to prepare a complete list of all applicable threats, policies and assumptions from the SPD. The only exception is any assumptions identifying threats that can be discounted or ignored. These do not generate security objectives, and therefore these assumptions, and their related threats, should be excluded.

The threats, policies and assumptions should be separated by type:

- Those relating to the non-IT operational environment;
- Those relating to the IT operational environment;
- Those related to TOE functionality;
- Those related to TOE development and TOE assurance.

This is usually easier than it might appear: a policy requiring physical controls can only apply to the non-IT environment; a threat concerning product design belongs to TOE development and assurance. Note that assumptions can only apply to the IT and non-IT operational environments. Where a policy or requirement appears to span several areas, it should be subdivided and one part assigned to each.
For example, a threat T.EAVESDROP might be split into two:

- T.EAVESDROP (communications), assigned to the IT operational environment;
- T.EAVESDROP (internal), assigned to TOE functionality.

If in doubt, split the policy or threat concerned into multiple areas. Unnecessary ones can be deleted later.

### 9.3 Objectives for the non-IT Operational Environment

The first step in identifying objectives for the non-IT operational environment is to take all the assumptions assigned to the non-IT operational environment and create corresponding objectives. Then devise and add any further objectives necessary to meet aspects of threats and policies assigned to the non-IT operational environment.

Environmental objectives are not analysed further within the PP and ST, or during evaluation, so there is little point in identifying commonality, generalisation, overlap etc., provided that the objectives are clear, and clearly defined.

Satisfying the non-IT operational environment objectives will be the responsibility of the organisation that uses the system or product in question. It is very important to check at this stage with the people responsible for system operation (or the marketing department in the case of products) to ensure that these objectives are realistic and achievable. If not, it is better to know of the problems now rather than later, while the objectives can still be altered or the threats and policies handled in different ways.

Environmental objectives are often given identifying names starting with “OE.” rather than “O.” This helps to make a clear distinction from TOE objectives. They should be clearly worded to indicate that the measures implementing the objective will be procedural or physical; if necessary state “the non-IT environment” explicitly in the description of the objective.

Environment objectives derived from assumptions are best worded unchanged from the assumption wording, i.e. as factual statements. For example:

OE.RESIDUAL  Magnetic media are degaussed or shredded prior to final disposal.

Objectives derived from threats and policies should be worded as requirements. For example:

OE.AUD_REVIEW  Operations staff will review audit trails for exceptions and unusual patterns of activity at regular intervals.

Most non-IT operational environment objectives will be derived from assumptions. Objectives that are derived from consideration of threats alone may indicate missing assumptions from the SPD specification. Check the SPD, and revise if necessary.
For convenience, single objectives can be defined that cover several related assumptions, or an assumption and related threats, or policies and related threats. It is worth combining such elements together if the overall result is clearer. If not, do not bother.

Identifying suitable wording for non-IT operational environment objectives that implement policies or counter threats is usually straightforward. If not, the categorisation techniques used for TOE objectives in section 9.5 below can be used.

### 9.4 Objectives for the IT Operational Environment

The techniques used to identify and specify the objectives for the IT operational environment are identical to those for the non-IT objectives described in 9.3 above. However, it is important to keep them separate from the non-IT objectives, as IT environment objectives could become TOE objectives if the TOE boundary changes.

Objectives for the IT operational environment should also be identified by giving them names that start with “OE.” Similarly, they should include “the IT environment” within the description, or otherwise make it clear that they will be implemented by technical means outside the TOE.

According to CC Version 2.2, security requirements for the IT environment can be specified for these objectives, if it is considered appropriate. This is not permitted in CC Trial Use Version 2.4. This Report recommends that this is done if and only if there is a specific and essential reason to do it, such as if a particular SFR from CC Part 2 is mandated by policies, or to enable a check on suitability of existing products to meet the relevant objectives by comparison of requirements with existing PPs or STs. In such cases, it is worth making the objectives for the IT operational environment as precise as possible, so that only the minimum areas need to be expanded into requirements. Other objectives can be left unexpanded.

There are other techniques, such as application notes, that can be used to record constraints on the implementation of objectives for the IT operational environment, and these are to be preferred to defining SFRs.

It should be noted, however, the PPST Guide contradicts the guidance above and suggests that requirements are always defined for the IT environment unless there is a justification for doing otherwise.

In a composite system, objectives for the IT environment of one domain will become objectives for the TOEs of other domains. Such objectives should be carefully worded, to ensure that the correspondence can easily be identified.

### 9.5 Objectives Related to TOE Functionality

The TOE security objectives are the most important and often the most difficult objectives to express well. They are used to generate SFRs from CC Part 2 functional
components. It is therefore important that they are well worded and clear in their
intention.

The first step in defining these objectives should be to reorder the list of applicable
threats and policies assigned to TOE functionality at the start of this step in order to
place related threats and policies together. There should be no assumptions relating to
TOE functionality, as assumptions are only made about the environment. If any
assumptions have been assigned to this heading, investigate and fix.

The best form of grouping for a particular PP or ST will depend on the nature of the
related TOE. However, it will be helpful later when generating SFRs if the grouping is
related to the structure of CC Part 2.

A good suggestion, which will work for many TOEs, is the following:

- TOE Security Behaviour (data protection, access control).
- User Binding (identification, authentication).
- Communication (TOE to users, other parts of TOE).
- TOE Self Protection (Reference Monitor etc.).
- Audit (event logging, analysis).
- Resource Management (finite constraints).
- Other Functions (anything not falling easily under these headings – e.g. trusted
time source, random number generation).

There is a deliberate close relationship between these suggested headings and the major
CC functional classes (this applies both to CC Version 2.2 and the proposals for CC
Version 3.0). The remaining CC classes mainly support parts of these groups; for
example, the FTA and FTP classes relate to TOE-user communication; FCS mainly
relates to communication; FPR mainly provides user binding without identification;
FMT mainly concerns security behaviour.

It should be obvious from the description of the security problem in the related SPD
what are the major security functions for a TOE, and the first step in defining objectives
should be to write down a simple definition of the service required in each of the above
areas needed to meet this overall requirement. Some areas may not be relevant; these
generate no definitions at this stage.

This list of services should then be compared against the list of threats and policies. For
each service, decide which threats and policies are relevant. At the end, put any threats
and policies remaining under the “other” service.

Next, divide the threats and policies associated with each service into general and
specific requirements. General requirements should apply to all aspects of the service
definition, specific requirements to special cases.

Finally, reword the service definition into a positive statement that addresses the general
requirement. This becomes the main objective for that service. Reword each specific
requirement into a related but separate subordinate objective for that service.
Threats can be countered by an objective that stops the threat by removing or blocking one of its necessary components. Examples of this are removing the ability of the threat agent to execute the adverse action, moving, changing or protecting the asset so that the adverse action is no longer applicable, or eliminating the threat agent (e.g. by introducing an environmental objective for physical access controls). Threats can also be handled indirectly. Examples of this are enforcing accountability through auditing actions, better training to stop accidental user errors, taking frequent backups so that lost or damaged assets can be easily restored.

Not all threats can be prevented. Sometimes the best course of action is to detect a related incident, and generate an alarm or audit log entry. This type of design decision will have to be made at this time. When detection is chosen as the response, this will generate the need for an audit service to respond to incidents.

During the specification process it may be necessary to reassign threats and policies. As services become better defined, particular threats or policies may fit more readily under a subordinate objective rather than the main objective or vice versa, or they may even fit better as part of another service. The process often identifies related objectives for the operational environment that have previously been missed; for example, there will be a need for administrators to respond to alarms, if alarms are chosen as the response to a particular threat. In some cases, design decisions may even move protection for particular threats or policies from the TOE objectives to the operational environment completely, or vice versa. These changes are to be expected; it will be necessary to iterate several times until a clear list of objectives is obtained covering all areas.

As well as expressing general protection requirements (linking directly to a main objective), policies in particular are sometimes used to constrain the nature of the associated technical solution. This type of constraint should be expressed as a subordinate objective, linked to the general requirement.

Some threats will link directly to a specific subordinate objective that counters that threat and no other. In this case, word the objective to directly reflect its source. This will ease later traceability – both in the rationale linking objectives back to the SPD, and for the understanding of readers. There is a useful table in Appendix B.7 of the PPST Guide that suggests suitable objectives for different types of asset/threat combinations.

A subordinate objective may address several threats and policies. For example, many PPs and STs have an object reuse objective as a subordinate objective in the area of resource management. This is worth separating out from other aspects of resource management as there is generally little overlap in terms of the threats addressed. However, there is no need to divide the objective further by the different types of resources that might need to be cleared – although different types of resources might be handled in different ways e.g. some threats to RAM do not apply to magnetic media. The distinction will become clear at the security requirements specification stage, when different SFRs will be selected as mechanisms for different resources.

A further useful distinction in defining subordinate objectives is by the type of control required. Controls can be preventative (stop an incident taking place), detective (recognise an incident has taken place) or corrective (fix the consequences of an incident). It is worth having different subordinate objectives for each type if the
treatment of threats or policies needs actions of more than one of these types in response. This is often the case if the description of the security problem requires defence in depth, or if the main objective for a service will only reduce or mitigate a threat rather than blocking it. Section 9.2 of the PPST Guide has a good explanation of selecting objectives based on these different types of controls.

Do not expect one to one correspondence between objectives and threats or policies. Often a main objective required to handle a policy will also counter many of the threats related to that service. Also, threats and policies may have to handled differently for different types of asset, and need different subordinate objectives for each asset type.

There are other techniques that can be used to identify security objectives. A simple approach, which can work well for small SPDs, is to simply to generate one objective per threat or policy, reflecting its wording, and with substitutions for specific assets, threat agents etc., if these are not clear from the wording of the related threat or policy in the SPD.

TOE objectives are generally given identifying names starting with “O.” rather than “OE.” to distinguish them from environmental objectives. They should be clearly worded to indicate that the measures implementing the objective will be part of and enforced by the TOE.

TOE objectives are sometimes worded to start “the TSF must” or “the system must”. The TSF is that part of the TOE that implements the SFRs. This distinction is made for practical reasons, to reduce the amount of the TOE that has to be examined during evaluation. The use of the term “TSF” is therefore strictly correct; for any objective, that part of the TOE that implements it must be part of the TSF. However, this is somewhat a circular argument and also a little confusing as these objectives are usually referred to as “TOE security objectives”, not “TSF security objectives”. System is a bad choice. It could be interpreted to include objectives implemented by the operational environment. If this is intended, it is much better to say “the TOE or its environment”. Note that design decisions must separate such objectives into objectives for the TOE and for its environment before the objectives are finalised.

9.6 Objectives for TOE Development and TOE Assurance

This is a confusing heading: the CC does not explain assurance and the development process very well. It should cover objectives for how the TOE is developed. These generate the SARs for the evaluation. It should also cover any objectives that are set for the environment in which the TOE is developed, although this is outside the scope of CC Version 2.2 evaluation.

In CC Version 2.2, these objectives are combined with the TOE functional objectives under the general heading of TOE security objectives.

In CC Trial Use Version 2.4, these objectives are referred to as development environment objectives, and listed separately, although most will refer to the development of the TOE, not the actual development environment.
In practice, development objectives are straightforward to define. There is usually an EAL with minor augmentations, chosen for cost/marketing/compatibility reasons rather than by analysis. A proper analysis is possible but rarely done.

The required EAL and augmentations should be defined as a policy, and reflected directly in a single development objective with matching wording.

It is recommended that development objectives are identified by names beginning with “OD.”, although this is unnecessary if there is a single objective and it will be included under a common CC Version 2.2 heading of “TOE Objectives”.

Where there are explicit objectives that relate to development or the development environment, care is need to ensure that they can be checked by evaluators. For example:

**OD.DEV_TRUST** The TOE is developed by permanent employees of the company that have been approved as trustworthy by the personnel department.

This would normally be derived from an assumption and therefore have to be considered as an environmental objective, and so out of scope of evaluation. But it is actually a matter of fact whether it is true or not, and so it could be represented as an explicitly stated security assurance requirement and checked by the evaluators. Of course, evaluators would not necessarily be appropriate people to check this assertion, and any PP or ST including such a requirement would not be accepted under current mutual recognition arrangements.

### 9.7 Producing the Objectives Rationale

The final step in defining the security objectives is to produce a rationale, tracing the objectives back to the threats, policies and assumptions in the SPD to show that they are all necessary, and also showing that all aspects of all threats, policies and assumptions in the SPD are covered by the objectives. This rationale is required by the CC, and checked in PP/ST validation.

A simple way to produce the rationale is to prepare tables of the relationships between the SPD elements and the objectives and vice versa, and check for any inconsistencies, gaps or overlaps. Where threats, policies or assumptions are handled by multiple objectives, there is usually a simple discriminant that can be attached to the SPD element to show which parts are countered by which objective – see 9.2 above. Including this in the table will make the mapping much clearer and easier to understand.

Remember that assumptions discounting threats apply to the SPD alone and should not appear in the rationale.
10 SELECTING SECURITY REQUIREMENTS

10.1 Introduction

The identification of security requirements is primarily a selection rather than definition process, and thus is usually rather more systematic and straightforward than previous steps in SPD or OP production. There is a lot of helpful and practical advice in the PPST Guide.

The important characteristic of security requirements is that they have to be evaluatable, i.e. concrete. This means that general statements in objectives and the SPD have to be broken down into specific requirements that can be tested. For example, statements in objectives that apply to all assets may have to be broken into separate technology-specific SFRs about each type of asset, or requirements for the existence of processes broken into individual actions e.g. an objective for user identification and authentication broken down into separate requirements covering user registration, logon, account maintenance, etc.

It is only necessary to define security requirements for the TOE objectives and for TOE development. Objectives for the IT and non-IT operational environment are not used in evaluation and therefore do not need to be expanded into requirements. As described in section 9.4 above, it is sometimes useful to generate SFRs to match some IT operational environment objectives. This can be done using exactly the same technique as for TOE objectives.

Although TOE objectives are mainly implemented by SFRs, sometimes they generate a small number of SARs. TOE development objectives only generate SARs. To simplify consistency checking, it is easier to process the TOE objectives first, so that any SARs required can be merged into development SARs.

10.2 Selecting Requirements for TOE Objectives

If the TOE objectives have been properly defined, they should be relatively independent of each other and can be addressed in turn. For each area, start with the main objective and then deal with the supporting objectives. Start with the most important area first. Leave any audit and management main objectives until last, as other objectives often generate additional requirements for these services.

If the methodology of section 10.2 above has been used, for each objective it will be easy to identify a small number of relevant CC Part 2 functional classes. These should be examined to identify the functional families which could provide security measures to meet the objective; the family behaviour section of the introduction to each family will help to do this. Within each family, the levelling information will identify the purpose of each component or hierarchy of components within the class. It is then a matter of deciding which components should be selected to meet the objectives. All components within relevant families should be considered. Where there is a hierarchy of components, the hierarchy as a whole should be considered for relevance first, and then, if relevant, the most appropriate member of the hierarchy selected.
Sometimes a single component will provide the necessary functionality to meet all of an objective; it is more common to need the combination of several components working together, usually selected from a small number of related families.

Once a component or components have been selected, all operations must be completed to generate the finished SFRs. There may be several different cases or requirements inherent in an objective that cannot be handled by a single instance of a component once completed. In this case, the completion process must be iterated several times on the component to generate multiple SFRs based on that one component.

This selection process is best illustrated by an example. Suppose there is a policy concerning password length:

_P.PASS_LEN_ All passwords will be not less than 8 characters long and include at least one numeric character.

This will result in a subordinate objective to the general authentication or user binding objective something along the following lines:

_O.PASS_LEN_ All passwords used to authenticate users must be not less than 8 characters long and include at least one numeric character.

This objective clearly falls within the scope of the FIA functionality class, and inspection shows that there are only two families that could contain relevant components, FIA_SOS Specification of secrets and FIA_UAU User authentication. Closer examination shows that there is only one suitable component, FIA_SOS.1 Verification of secrets. FIA_UAU.5 Multiple authentication mechanisms, also deals with the way authentication mechanisms work, but not with the characteristics of each mechanism and thus is not appropriate.

FIA_SOS.1 states:

_FIA_SOS.1.1_ The TSF shall provide a mechanism to verify that secrets meet [assignment: a defined quality metric].

To complete the SFR, it is necessary to insert the password constraints into the assignment, and for readability, to replace the generic term “secrets” with the actual secret “passwords used for user authentication”.

_FIA_SOS.1.1_ The TSF shall provide a mechanism to verify that passwords used for user authentication meet the following criteria: all passwords are not less than 8 characters long and include at least one numeric character.

There will need to be related SFRs to ensure that users do authenticate to the TOE, but we can assume that these will be generated by the related general user binding objective.

In passing, we note that this was a very poor policy to choose – it does not prevent passwords made up of all numerics that would be easily guessable if observed. This weakness has been faithfully reproduced in the objective and the derived functional
requirement. The process of SPD and OP preparation must always look out for ambiguities or possible errors in specifications and requirements.

There may be several suitable components, possibly even from different families, that could meet a particular objective or part of an objective. Deciding which components to use will often force the design of the systems or product down a particular path. Where a genuine choice exists, consult other system designers to establish which alternative is best overall.

Objectives can be interlinked. SFRs required for one objective can provide functionality that can be reused to implement other services. Existing functionality should be reused if possible; include a cross-reference to the other objective for use in constructing the requirements rationale. Not all duplications are obvious, so a further check will be made later, when all SFRs have been generated.

Many components suggest audit or management actions to be included in the PP or ST if they are selected. Appropriate actions should be identified and listed, and covered by the specification of the audit and management objectives when these are processed. Not all PPs or STs will have audit or management objectives. In these cases, audit or management actions must either be eliminated or handled in some other way. This normally requires an explicitly defined component (such as “stop on error”).

In the case of the password example used above, FIA_SOS.1 has the following suggested management function:

*The management of the metric used to verify the secrets.*

FIA_SOS.1 also has the following suggested audit functions:

*Minimal: Rejection by the TSF of any tested secret;*

*Basic: Rejection or acceptance by the TSF of any tested secret;*

*Detailed: Identification of any changes to the defined quality metrics.*

In the case of this example, the rules governing acceptable passwords are fixed by the policy and therefore do not require management. The minimal audit requirement would be to log rejection of passwords selected by a user that failed to meet the given criteria, and the basic requirement to log both rejection and acceptance of passwords. The detailed requirement is inapplicable as the criteria cannot change. In practice, there is not a lot of value in logging password changes for conformance to acceptable password criteria, and this audit requirement might be argued away if password changes were logged for other purposes (such as establishing a period of validity).

Once all of the objectives have been addressed, it is necessary to perform a dependency analysis of the generated SFRs. Many components are identified in CC Part 2 as having dependencies upon other components. In many cases these will have already been satisfied because the relevant components will have been selected as part of the coverage required to meet the relevant objective. However, all SFRs should be checked for unsatisfied dependencies and any unsatisfied dependencies resolved.
In some cases this will mean that additional SFRs have to be added (these are called supporting SFRs in the PPST Guide). However, often the dependency is not applicable due to the nature of the way the component has been used, or because the necessary component is present as part of the SFRs generated for another objective. These cases should be noted and explained so that the necessary justification can be included in the requirements rationale (see 11.3 below).

A check should also be made for any components that have been completed more than once into identical or nearly identical SFRs. This can happen if the same basic functionality is needed to meet more than one objective, and the overlap was not detected whilst generating the SFRs. If such cases are found, the duplication should be eliminated; it may be worthwhile to reword one or more of the related objectives to remove the cause of the duplication.

TOE objectives normally generate SFRs. However, in some cases SARs are required, either explicitly or as a result of dependency analysis. This is not a problem; the division between functional components and assurance components in CC Version 2.2 is sometimes blurred, with objectives requiring both functionality and assurance measures. If a TOE objective generates only assurance requirements, then it is almost certainly a TOE development objective, and has been mis-categorised. It is expected that CC Version 3.0 will have a clearer distinction between functionality and assurance requirements.

Sometimes no components in CC Part 2 or Part 3 will be suitable to meet a particular objective. In this case new components will need to be defined; these are called extended requirements (see 10.4). Such explicitly stated requirements are rarely necessary in practice, since most classes contain at least one very general component that can be completed to cover almost anything. Do not be afraid to use explicitly stated requirements if they are necessary, but recognise that a full definition and justification will be required as part of the PP or ST.

In very rare cases, an objective may be impossible to achieve by any combination of components, including explicitly defined components. This represents an impossible requirement. The objectives will have to be re-specified, normally to permit satisfaction of the SPD in a different way using different objectives. The SPD may need to be changed as a result.

Sometimes SFRs are generated, but later rejected for external reasons such as cost of implementation. In these cases, alternative SFRs will have to be selected or the objectives will changed. It is possible that objectives will then be reassigned to the environment rather than the TOE, with a follow-on change required to the SPD that generated them.

10.3 Selecting Requirements for TOE Development

Selecting the security assurance requirements corresponding to the TOE development and development environment objectives is usually a much simpler process than selecting requirements to meet the TOE functional objectives.
The development objectives are usually defined in assurance terms, as an EAL, possibly with minor augmentations from CC Part 3 (see 9.6 above). Since an EAL is a predefined package of SARs, all that is required is to list its components and merge in any augmentations or replacements that have been specified in the objectives.

If a full analysis is required, the same techniques can be applied as given in 10.2 above. The analysis is slightly simpler as development objectives will never generate SFRs, even as dependencies. Otherwise the approach is the same as for TOE objectives.

10.4 Defining Extended Requirements

Where no predefined suitable predefined components exist in CC Part 2 or CC Part 3, it is permissible to define new components. Be aware, however, that explicitly defined assurance components will usually prevent mutual recognition of evaluation results based on any PP or ST containing them.

If explicitly defined components are required, they should be defined in the same style as used for CC part 2 and CC part 3 components, and, if possible, linked into existing CC families or classes. The definition must be included in the PP or ST as part of the Objectives Package that uses the extended component.

The PPST Guide provides good advice on defining explicitly defined components (see [7], sections 10.2.8 and 10.3.3).

10.5 Practical Advice

The PPST Guide has a very good explanation of the steps involved in generating SFRs from functional components – completing permitted operations (assignment, iteration, selection), satisfying dependencies, and identifying and handling management and audit requirements.

The PPST Guide suggests that because of the formulaic and lengthy nature of requirements, they are documented in an annex to the PP or ST. Although this is a sensible suggestion, few PPs or STs seem to follow it in practice, perhaps because of the degree of cross-referencing to and from the body of the text that seems to follow.

It will be found that some functional requirements can be satisfied using CC Part 2 components in more than one way. Some very specific situations have components defined that are instances of more general requirements. An example of this is FIA_AFL.1 Authentication failure handling. This component could also be expressed as a combination of FAU_GEN.1 Audit data generation, FAU_SAA.1 Potential violation analysis, and FAU_ARP.1 Security alarms. FIA_AFL.1 Authentication failure handling is therefore logically redundant, but it was included in CC Part 2 because it very clearly expresses a specific requirement that is often used and it provides a much clearer description that the combination of the three other components.

---

1 This example was taken from Part 1 of CC Trial Use Version 2.4 [6].
Existing STs, requirements packages and PPs can be used as a source of previously generated sets of SFRs and SARs to meet particular objectives. This can save time, reduce the risk of introducing errors, and ensure compatibility. The US Consistency Instruction Manuals are a good example of the use of standardised requirements to meet the same national policies across a wide range of requirements.

The formal language of SFRs and SARs sometimes causes the reason for including a particular requirement to be lost or not readily deducible from the wording of the completed component. Use application notes liberally to explain why requirements are present. Do not put explanations just in the requirements rationale: other people may not read the rationale.
11 FINALISING OBJECTIVE PACKAGES

11.1 Introduction

Having generated all the SFRs and SARs required to satisfy the objectives, the final step in OP production is to assign the objectives and related requirements to suitable packages, and to create the necessary rationales to show complete coverage of the associated SPD.

11.2 Selecting Package Contents

Identification of suitable package boundaries will be driven by two factors – the architecture of the TOE, and the relationships between objectives. There is no point introducing multiple OPs unless there are clear divisions in functionality that will reduce the scope of individual evaluation tasks.

Every functional objective assigned to a particular OP will require all its associated SFRs to be included. These SFRs may have dependencies on SFRs in other packages. Each such SFR should be checked to ensure that the relevant service will be available, not blocked by architectural boundaries or limited by practical issues such as bandwidth constraints, etc. If necessary the supporting SFRs will have to be duplicated in both packages.

SARs generated from development objectives normally apply to all OPs, but this need not be the case if different parts of the TOE corresponding to different OPs are manufactured in different environments. However, any SARs generated from TOE requirements should only be included in the OPs corresponding to the relevant functional objectives.

In general, specifying separate OPs for the operational environment of the TOE is not worthwhile; it is better to include these objectives in related TOE OPs.

Separate IT environmental OP packages are useful if they match OPs for other TOEs and can be used to show compatibility in composition. Similarly, a separate non-IT environment OP can be useful if it become a baseline specification for the security operational procedures of a system.

11.3 Producing the Requirements Rationale

The requirements rationale must cover a number of topics. This rationale is required by the CC, and checked in PP/ST validation.

Firstly, the rationale must show that every objective is included in an OP, and there are no objectives in OPs that are not in the original list of objectives derived from analysis of the threats, policies and assumptions of the original SPD or SPDs.
During the course of requirement specification, objectives may have been amended to remove ambiguities or improve implementation. In these cases, the objectives rationale will have to be checked and if necessary updated.

The second task is to show for each OP that its objectives are implemented by the set of selected SFRs and SARs, and any dependencies are satisfied, not required or satisfied by other OPs in a way that is technically feasible. In addition, a check must be made to show that all the selected SFRs and SARs are necessary to meet the objectives of the OP.

Finally the rationale must show that any SARs derived from TOE requirements are consistent and compatible with the SARs for TOE development.

As for other rationales, a tabular approach is often useful.

If an objective satisfies several threats or policies, individual SFRs may be required to handle all the threats and policies, or may be linkable to one specific threat or policy. The US Consistency Instruction Manuals suggest making these relationships explicit by putting details of the relevant SFRs in the objectives rationale. This will make tracing from threats and policies to SFRs easy, but complicates the rationales, and thus checking rationale consistency. It is probably better to break down threats and policies in the objectives rationale to show the correspondence to particular objectives, and then use the requirements rationale to link objectives to specific SFRs.

PPs contain both objectives and requirements. If a PP to which an ST wishes to claim compliance was specified in terms of incompatible assets and/or threats or policies, showing compliance in the rationale can be difficult. It is necessary to include all the objectives, SFRs and SARs from the PP in the ST, and then trace back to both the PP and any other threats and policies. CC V2.4 tries to address this issue, by enabling a degree of flexibility in compliance with PPs. If possible, the easiest route is to go for exact conformance to a PP (i.e. ensuring that there are exactly the same threats/policies/assets/objectives/SFRs). Otherwise the best technique is to incorporate the text of the PP in the ST and then rework and revise the various rationales (an approach similar to CC V2.4 strict conformance). Alternatively, a PP can be used just as a source of compatible pre-selected SFRs and SARs to be linked back to standardised objectives (similar to CC V2.4 demonstrable conformance).
12 MANDATED DEVELOPER ACTIONS

12.1 Introduction

This chapter identifies all the mandatory requirements that must be satisfied when preparing a PP or ST in accordance with CC Version 2.2. CC Version 2.2 is identical to Version 2.1, except that it incorporates finalised interpretations on the meaning of Version 2.1. It is therefore a more stable baseline for discussion.

This chapter also covers the proposed changes incorporated in Trial Use Version 2.4. Many of the remaining problems in Version 2.1 are addressed by this version. In many cases, CC Trial Use Version 2.4 requirements are very similar to Version 2.1 and 2.2 requirements, but have been reworded to improve clarity or indicate what evidence is required more precisely. Developers of PPs and STs should take account of the Version 2.4 changes to wording, as indicated in this chapter, as an indication of the generally accepted interpretation of the CC’s original intentions. There are a very small number of new features in Trial Version 2.4. These are not discussed in this chapter, since these ideas are only now beginning to be trialled in practice.

This chapter is structured as follows. The main elements of an ST are dealt with in separate sections:

- ST Introduction
- PP Claims
- SPD (Security Environment)
- Security Objectives
- Security Requirements
- TOE Summary Specification

A final section deals with the differences between PPs and STs.

12.2 ST Introduction

The developer of a Security Target must provide an ST introduction. This identifies the ST and puts it into context. It is covered by two criteria families, ASE_INT and ASE_DES. There are two developer action requirements, one per family:

*ASE_INT.1.1D The developer shall provide an ST introduction as part of the ST.*

*ASE_DES.1.1D The developer shall provide a TOE description as part of the ST.*

Although the TOE description is treated as a separate requirement in CC Version 2.2, it is closely linked to the ST Introduction, and in Trial Use Version 2.4 the two have been merged together. ST developers are therefore recommended to treat the two requirements as a single unit.

The contents of the ST Introduction, including TOE description, are covered by four requirements:
ASE_INT.1.1C  The ST introduction shall contain an ST identification that provides the labelling and descriptive information necessary to control and identify the ST and the TOE to which it refers.

Trial Use Version 2.4 clarifies this requirement. It is sufficient to provide two references, one that uniquely identifies the ST, and one that uniquely identifies (or will identify) the TOE to which it refers.

ASE_INT.1.2C  The ST introduction shall contain an ST overview which summarises the ST in narrative form.

This is also clarified by Trial Use Version 2.4. It is sufficient to summarise the intended use and major security features of the TOE, together with any non-TOE hardware/software/firmware required by the TOE. It is important to identify any IT functions or facilities required by the TOE that provide security features or properties required by the TOE. These will appear as technical security features in the assertions section of the SPD.

ASE_INT.1.3C  The ST introduction shall contain a CC conformance claim that states any evaluable claim of CC conformance for the TOE.

This requirement only refers to criteria conformance; conformance with PPs is handled in a separate section. In CC Trial Use Version 2.4, these two aspects of conformance are dealt with in a single section of the ST.

CC Trial Use Version 2.4 clarifies what the claim of conformance should cover. It should identify the version of the CC to which the ST and the TOE claim conformance, describe the ST’s conformance to Part 2 as either CC Part 2 conformant or CC Part 2 extended, and its conformance to Part 3 as either CC Part 3 conformant or CC Part 3 extended. The CC conformance claim must be consistent with any explicitly stated security component definitions provided in the security requirements section; i.e. if explicitly stated functional components are defined, there cannot be a claim of CC Part 2 conformance.

ASE_DES.1.1C  The TOE description shall describe the product or system type, and the scope and boundaries of the TOE in general terms both in a physical and a logical way.

There is no predefined set of product and system types. A simple one or two word description such as firewall, VPN-firewall, smartcard, crypto-modem, web server, or LAN with web server and database will be sufficient. The second part of this requirement asks for two conceptual descriptions of the TOE, a description of its physical scope and boundaries, and then a description of its logical scope and boundaries. The TOE description should be kept short; a fuller explanation appears later within the TOE Summary Specification.

12.3 PP Claims

The next section of a Security Target is the PP claims section. It identifies any Protection Profiles to which the ST claims compliance, together with the necessary
supporting information to justify the claims. If there are no claims of PP compliance, the whole section is omitted. It is covered in a single criteria family, ASE_PPC. There are two developer action requirements:

ASE_PPC.1.1D The developer shall provide any PP claims as part of the ST.

ASE_PPC.1.2D The developer shall provide the PP claims rationale for each provided PP claim.

The statement of PP claims has three content requirements:

ASE_PPC.1.1C Each PP claim shall identify the PP for which compliance is being claimed, including qualifications needed for that claim.

Although it is clear that this requirement only permits compliance to be claimed to complete PPs, in practice it is often necessary to claim compliance to functional or assurance packages as well. This is explicitly permitted in CC Trial Use Version 2.4. A particularly common case is to claim compliance to the functional components of a PP, but at a higher or lower EAL. This could be regarded as a qualification, as permitted by this requirement. The more normal qualification is to identify any uncompleted operations in the PP that are fully completed within the ST, but this is explicitly requested in the next content requirement:

ASE_PPC.1.2C Each PP claim shall identify the IT security requirements statements that satisfy the permitted operations of the PP or otherwise further qualify the PP requirements.

The final content requirement is rather curious:

ASE_PPC.1.3C Each PP claim shall identify security objectives and IT security requirements statements contained in the ST that are in addition to those contained in the PP.

This is a redundant requirement. The necessary information is already contained in the security objectives and security requirements sections of the ST. The best approach is to just include two cross-references, one to each section of the ST. This requirement has been removed from Trial Use Version 2.4.

Within CC Version 2.2, there are no content requirements for the conformance claims rationale. However, a rationale is necessary. In CC Trial Use Version 2.4, its contents are defined as follows:

V2.4_ASE_CCL.1.7C The conformance claims rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

V2.4_ASE_CCL.1.8C The conformance claims rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.
V2.4_ASE_CCL.1.9C The conformance claims rationale shall demonstrate that the statement of objectives is consistent with the statement of objectives in the PPs for which conformance is being claimed.

V2.4_ASE_CCL.1.10C The conformance claims rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

V2.4_ASE_CCL.1.11C The conformance claims rationale shall demonstrate that all operations of the security requirements that were taken from a PP are completed consistently with the respective PP.

With the exception of V2.4_ASE_CCL.1.7C, these are all logically necessary. The proof that permitted operations left open by the PP have been correctly completed is particularly important.

V2.4_ASE_CCL.1.7C is more difficult to justify; there is no defined set of TOE types and thus consistency would be entirely subjective.

CC Trial Use Version 2.4 also contains equivalent requirements for conformance to a package rather than a PP.

The CC permits PP contents to be referenced or restated in conformance claims. If differences from or additions to the published PP are small, it is best to identify the differences. If differences are major, the PP should be restated – and the value of the claiming PP conformance carefully considered.

The CC permits an ST to claim conformance to multiple PPs. This is rare in practice – PPs have to be complete and self-consistent. When combined, there is likely to be a major overlap of requirements between PPs, and possibly even incompatible requirements.

12.4 SPD (Security Environment)

The developer of a Security Target must provide a description of the security problem addressed by the ST. In CC Version 2.2 this is called the Statement of Security Environment. This name is confusing since it must cover both the TOE and the TOE environment. In CC Version 2.4, and throughout this Report, the name Security Problem Definition is used instead.

The SPD is covered by one criteria family, ASE_ENV. There is one developer action requirement:

ASE_ENV.1.1D The developer shall provide a statement of TOE security environment as part of the ST.

As described elsewhere in this Report, the SPD is defined in terms of threats, policies and assumptions. There are three content requirements.
The first content requirement covers assumptions:

*ASE_ENV.1.1C*  The statement of TOE security environment shall identify and explain any assumptions about the intended usage of the TOE and the environment of use of the TOE.

Part 1 of CC Version 2.2 clarifies that “intended usage” is supposed to cover security aspects of the environment such as intended application, potential asset value and possible limitations of use. This information could also be presented in the TOE description that forms part of the TOE Introduction, and for ease of reference and consistency it is suggested that such information is presented only in the TOE description. This part of the requirement has been dropped from Trial Use Version 2.4.

“Environment of use” refers to the operational environment, although this Report demonstrates that it may also be necessary to include assumptions about the development environment (for example, that only trustworthy staff will be used in developing the TOE).

It is customary to list assumptions under three headings: physical, personnel and connectivity, although this is not mandated by the CC. This Report suggests that a more suitable set of headings are:

- physical protection;
- personnel and procedures;
- technical functionality outside the TOE.

In particular, it is often necessary to make assumptions about technical controls in the operational environment of the TOE that are unrelated to connectivity, a term specifically defined in Part 1 of the CC.

The second content requirement concerns threats:

*ASE_ENV.1.2C*  The statement of TOE security environment shall identify and explain any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment.

This is a poorly worded requirement. Part 1 of CC Version 2.2 requires that threats are described in terms of an identified threat agent, an attack, and the asset that is the subject of the attack. It also states that not all possible threats that might be encountered in the environment need to be listed, only those which are relevant for secure TOE operation. This is not consistent with the actual requirement above.

This characterisation of threats is inadequate. In order to assess whether security controls are necessary and adequate, it is necessary to know how an attack might damage an asset; a successful attack that does not damage an asset in any way is unfortunate but not of great consequence. CC Trial Use Version 2.4 has a much better characterisation for threats:

*V2.4_ASE_SPD.1.1C*  The security problem definition shall describe the threats.
V2.4_ASE_SPD.1.2C All threats shall be described in terms of a threat agent, an asset, and an adverse action.

It is recommended that this formulation is used in all STs, unless the threats were identified from a specific external activity, such as a formal risk analysis. If the external statement of threats is clear and provides sufficient information to derive the security objectives, then it is better to use the statement of threats unchanged, rather than inventing details such as threat agents. This can be explained and justified in the rationale or by application notes.

“Asset” should also be understood to permit “type of asset”, as it is pointless to itemise different assets with identical characteristics and protection requirements. It is only necessary to describe their relevant characteristics.

The final content requirement deals with Organisational Security Policies (OSPs):

ASE_ENV.1.3C The statement of TOE security environment shall identify and explain any organisational security policies with which the TOE must comply.

This is another poorly worded requirement. Policy is an overused word in the Common Criteria. Policy in the sense of OSPs is completely distinct from use of the word in the sense of TOE Security Policy (TSP). The TSP defines the rules by which the TOE governs access to its resources, and thus all information and services controlled by the TOE.

It should be noted that OSPs may be more than organisation-wide (for example, where particular security functions are mandated by national law), but may also be applicable only to part of the organisation (for example, an OSP to provide protection against a particular threat may only apply to the system in question). The full name is therefore inappropriate. It is better just to call them “policies”.

Finally, a policy is a requirement to do something. It is therefore pointless to explain OSPs. No justification is needed without the context of the ST. Trial Use Version 2.4 has reworded and simplified this requirement to make this clear:

V2.4_ASE_SPD.1.3C The security problem definition shall describe the OSPs.

Note that there is no requirement in the CC to provide a rationale justifying the security problem definition; it is taken as axiomatic in ST evaluation. However, this Report recommends that a source rationale is included as part of the SPD. Over time, security needs will change. Provision of a rationale will make it much easier to assess the impact of changes on the ST, and, if necessary, to update it.

There is much more information about formulating SPDs in earlier chapters of this Report.
12.5 Security Objectives

Following the definition of the security problem, the next section of a Security Target is the Security Objectives section. This is defined as a concise statement of the intended response to the security problem. In practice, it is made up of a list of aspects of security where controls will be required. It is covered in a single criteria family, ASE_OBJ. There are two developer action requirements:

*ASE_OBJ.1.1D* The developer shall provide a statement of security objectives as part of the ST.

*ASE_OBJ.1.2D* The developer shall provide the security objectives rationale.

There are five content requirements. The first merely states that the statement of security objectives must define the security objectives:

*ASE_OBJ.1.1C* The statement of security objectives shall define the security objectives for the TOE and its environment.

It is not clear from this requirement that it is necessary to distinguish objectives for TOE from objectives for its environment, but in practice this is essential, as objectives for the environment are excluded from TOE evaluation. Also, within CC Version 2.2 the treatment of assurance objectives is confused. The CC requires objectives to be set for assurance in the TOE to be determined by evaluation. Since almost all such assurance comes from assessment of the way that the TOE was developed, these are best thought of as objectives for its development environment rather than the TOE.

CC Trial Use Version 2.4 expresses this clear by splitting this content requirement into three. It also replaces the word “define” by “describe”, which is more consistent with other content requirements:

*V2.4_ASE_OBJ.1.1C* The statement of security objectives shall describe the security objectives for the TOE.

*V2.4_ASE_OBJ.1.3C* The statement of security objectives shall describe the security objectives for the development environment.

*V2.4_ASE_OBJ.1.5C* The statement of security objectives shall describe the security objectives for the operational environment.

The second content requirement ensures that all TOE security objectives are necessary:

*ASE_OBJ.1.2C* The security objectives for the TOE shall be traced back to aspects of the identified threats to be countered by the TOE and/or organisational security policies to be met by the TOE.

This requirement is poorly worded. The tracing should form part of the security objectives rationale, but the requirement does not say this. Also, tracing should be performed separately for each security objective, to show that it is not superfluous. This
requirement has been reworded in CC Trial Use Version 2.4 to make these matters clear:

V2.4_ASE_OBJ.1.2C The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs met by that security objective.

The third requirement ensures that all environmental security objectives are necessary:

ASE_OBJ.1.3C The security objectives for the environment shall be traced back to aspects of identified threats not completely countered by the TOE and/or organizational security policies or assumptions not completely met by the TOE.

Note that assumptions are only permitted to be satisfied by environmental objectives. As already noted, there will be usually be security objectives for the development environment and development process for the TOE, and these will be need to be assigned to either the TOE objectives or the environment objectives, depending on whether they will be tested by evaluation or not.

This is clarified in CC Trial Use Version 2.4 by separating the development and environmental objectives:

V2.4_ASE_OBJ.1.4C The security objectives rationale shall trace each security objective for the development environment back to threats countered by that security objective and OSPs met by that security objective.

V2.4_ASE_OBJ.1.6C The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.

However, there is still a problem with V2.4_ASE_OBJ.1.4C; there may be security objectives for the development environment that must trace back to assumptions as well as threats or policies.

The final two content requirements ensure that the security objectives provide complete coverage of all threats, policies and assumptions:

ASE_OBJ.1.4C The security objectives rationale shall demonstrate that the stated security objectives are suitable to counter the identified threats to security.

ASE_OBJ.1.5C The security objectives rationale shall demonstrate that the stated security objectives are suitable to cover all of the identified organisational security policies and assumptions.

In it not obvious why threats are treated separately from policies and assumptions, and why there is no word “all” in ASE_OBJ.1.4C. These minor issues are resolved in the equivalent content requirements in CC Trial Use Version 2.4:
The security objectives rationale shall demonstrate that the security objectives counter all threats.

The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.

The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

The problem of assumptions about the development environment remains.

There is more information about formulating security objectives in earlier chapters of this Report.

12.6 Security Requirements

Following the Security Objectives, the lowest level of detail in a Security Target is the Security Requirements. This is intended to be a set of structured, testable factual statements that can be checked by an evaluator. Where possible, security requirements should be taken from Part 2 and Part 3 of the Common Criteria. However, it is possible to define extended requirements if none of the predefined ones are suitable. It is covered in two criteria families, ASE_REQ and ASE_ECD, covering the predefined and extended security requirements respectively. There are two developer action requirements for ASE_REQ:

**ASE_REQ.1.1D** The developer shall provide a statement of IT security requirements as part of the ST.

**ASE_REQ.1.2D** The developer shall provide the security requirements rationale.

These are duplicated word for word in ASE_SRE:

**ASE_SRE.1.1D** The developer shall provide a statement of IT security requirements as part of the ST.

**ASE_SRE.1.2D** The developer shall provide the security requirements rationale.

This duplication was thought necessary because ASE_SRE has different content requirements to ASE_REQ. In CC Trial Use Version 2.4 this duplication is handled differently. The statement of security requirements only appears in the equivalent to ASE_SRE, namely ASE_ECD, although there are still content requirements in ASE_REQ.

Security requirements in Part 2 and Part 3 take the form of component definitions. Each definition contains several requirement statements, called elements, which must be taken and used as a whole. In addition, the elements may contain operations, pieces of general text which must be completed when the definition is included within an ST.
For extended requirements, it is sensible to specify a component definition as well as the completed form included in the statement of security requirements. This is mandated in CC Trial Use Version 2.4:

*V2.4装配式ECD.1.2D The developer shall provide an extended components definition.*

There are twenty content requirements, thirteen for components taken from Part 2 and 3, and seven for extended components. This is therefore the most complex area of an ST from the point of view of specification requirements.

The first content requirement is that the statement of IT security requirements includes all the security requirements for the TOE:

*ASE_REQ.1.1C The statement of TOE security functional requirements shall identify the TOE security functional requirements drawn from CC Part 2 functional requirements components.*

*ASE_REQ.1.2C The statement of TOE security assurance requirements shall identify the TOE security assurance requirements drawn from CC Part 3 assurance requirements components.*

*ASE_SRE.1.1C All TOE security requirements that are explicitly stated without reference to the CC shall be identified.*

There is therefore no requirement to separate the explicitly stated functional requirements from the explicitly stated assurance requirements, but it would be pointless not to do so. CC Trial Use Version 2.4 does not bother to make this distinction, as it is obvious from naming conventions. The verb “identify” is unclear as to what it precisely means. In fact, to meet other content requirements all the security requirements must be written out in full, with operations completed. The names of the components will identify their source.

As well as the statement of TOE security assurance requirements, an Evaluation Assurance Level is normally selected:

*ASE_REQ.1.3C The statement of TOE security assurance requirements should include an Evaluation Assurance Level (EAL) as defined in CC Part 3.*

Note that this is an optional requirement; it uses the verb “should”, not “shall”. In practice, EALs are expected by users of the CC and an EAL will be specified. In CC Trial Use Version 2.4 this requirement forms part of the conformance claims section of the ST.

ASE_REQ.1.4C is discussed later in this section as part of ASE_REQ.1.12C. The next two content requirements concern security requirements for the IT environment:

*ASE_REQ.1.5C The ST shall, if appropriate, identify any security requirements for the IT environment.*
ASE_SRE.1.2C  All security requirements for the IT environment that are explicitly stated without reference to the CC shall be identified.

These requirements apply if there are assumptions in the Security Problem Definition concerning technical functionality outside the TOE. However, security requirements for the IT environment are not evaluated. There is therefore no value in expanding the relevant security objectives. There may be also dependencies of SFRs within the TOE that are satisfied by the IT environment (for example, a secure time source outside the TOE providing timestamps to satisfy the FAU_GEN.1 dependency on FPT_STM.1), but this can be explained more directly in the rationale for non-satisfaction of dependencies (see ASE_REQ.1.8C). There are a number of open Requests for Interpretation applicable to CC Version 2.2 that relate to this issue.

The best advice that can be given at present is not to provide requirements for any environmental objectives, and to state that the whole environment is treated as axiomatic and thus specification of security requirements for the IT environment is not appropriate.

This approach is consistent with CC Trial Use Version 2.4, where there is no requirement to specify security requirements for the IT environment.

The next content requirement is to complete all operations:

ASE_REQ.1.6C  Operations on IT security requirements included in the ST shall be identified and performed.

This is more clearly expressed in CC Trial Use Version 2.4:

V2.4_ASE_REQ.*.2C  The statement of security requirements shall identify all operations on the security requirements.

V2.4_ASE_REQ.*.3C  All assignment and selection operations shall be completed.

V2.4_ASE_REQ.*.4C  All operations shall be performed correctly.

All dependencies between components must either be satisfied or else explained away in the rationale:

ASE_REQ.1.7C  Dependencies among the IT security requirements included in the ST should be satisfied.

ASE_REQ.1.8C  The evidence shall justify why any non-satisfaction of dependencies is appropriate.

Once again, this is better expressed in CC Trial Use Version 2.4:

V2.4_ASE_REQ.2.5C  Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
The next set of content requirements concern strength of function:

*ASE_REQ.1.9C*  The ST shall include a statement of the minimum strength of function level for the TOE security functional requirements, either SOF-basic, SOF-medium or SOF-high, as appropriate.

*ASE_REQ.1.10C*  The statement of security requirements shall identify all security functional requirements for which an explicit strength of function claim is required, together with the explicit strength of function claim for each such security functional requirement.

*ASE_REQ.1.11C*  The security requirements rationale shall demonstrate that the minimum strength of function level for the ST together with any explicit strength of function claim is consistent with the security objectives for the TOE.

There is no requirement for a strength of function claim in CC Trial Use Version 2.4. The CCIMB took a decision to remove the requirement for a SOF-claim in the ST, as it is not appropriate to all TOEs. There are other issues relating to SOF-claims:

- SOF-related requirements are the only requirements in the ST which trace forward to the mechanism level of detail, which takes a substantial amount of developer effort to be expended on providing this in the ST;
- some of the developer information needed to provide this mechanism information may be of a confidential nature, and therefore not suited for a public document such as the ST.

In these circumstances, it is recommended that a minimum strength of function of SOF-basic is claimed covering all security functions in the TOE, and the choice is justified in the rationale as the default rating used where no specific higher strength requirements apply. Of course, if there is a clear reason for a more specific or higher rating, it can be specified and justified in the rationale.

The next set of content requirements concern the security requirements rationale. This must show that the security requirements meet the security objectives and are mutually consistent:

*ASE_REQ.1.12C*  The security requirements rationale shall demonstrate that the IT security requirements are suitable to meet the security objectives.

*ASE_REQ.1.13C*  The security requirements rationale shall demonstrate that the set of IT security requirements together forms a mutually supportive and internally consistent whole.

These requirements are better expressed in CC Trial Use Version 2.4 as requirements to trace every selected requirement back to a security objective, and to show that all objectives are covered:

*V2.4_ASE_REQ.2.6C*  The security requirements rationale shall trace each SFR back to the security objectives for the TOE.
V2.4_ASE_REQ.2.8C The security requirements rationale shall trace each SAR back to the security objectives for the development environment.

V2.4_ASE_REQ.2.7C The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.

V2.4_ASE_REQ.2.9C The security requirements rationale shall demonstrate that the SARs meet all security objectives for the development environment.

There is also an explicit requirement in CC Version 2.2 to show that the choice of TOE security assurance requirements is appropriate:

ASE_REQ.1.4C The evidence shall justify that the statement of TOE security assurance requirements is appropriate.

It is not clear what “appropriate” means in this context beyond what is covered by ASE_REQ.1.13C. There is no equivalent in CC Trial Use Version 2.4. It is therefore recommended that this requirement is ignored.

The remaining content requirements only apply to explicitly stated IT security requirements. The first is a requirement to explain in the rationale why explicitly stated security requirements were needed:

ASE_SRE.1.3C The evidence shall justify why the security requirements had to be explicitly stated.

It should be sufficient to state that no requirements in CC Part 2 and Part 3 were completely appropriate. This requirement has been removed from Trial Use Version 2.4.

The next content requirements mandate that the explicitly stated component definitions follow the structure of the CC components, are clearly expressed, and are evaluatable:

ASE_SRE.1.4C The explicitly stated IT security requirements shall use the CC requirements components, families and classes as a model for presentation.

ASE_SRE.1.6C The explicitly stated IT security requirements shall be clearly and unambiguously expressed.

ASE_SRE.1.5C The explicitly stated IT security requirements shall be measurable and state objective evaluation requirements such that compliance or noncompliance of a TOE can be determined and systematically demonstrated.

ASE_SRE.1.7C The security requirements rationale shall demonstrate that the assurance requirements are applicable and appropriate to support any explicitly stated TOE security functional requirements.

CC Trial Use Version 2.4 makes it clear that it is the component definitions, rather than their completed forms as actual security requirements, that should be based on CC components, and that the extended components should consist of measurable and
objective elements such that compliance or non-compliance to these elements can be demonstrated:

V2.4_ASE_ECD.1.2C  The extended components definition shall define an extended component for each extended security requirement.

V2.4_ASE_ECD.1.3C  The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

V2.4_ASE_ECD.1.4C  The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

V2.4_ASE_ECD.1.5C  The extended components shall consist of measurable and objective elements such that compliance or noncompliance to these elements can be demonstrated.

These V2.4 alternatives imply that there is no methodological problem evaluating extended functional components using existing security assurance requirements, provided that the new elements are measurable and objective, and that this is sufficient to demonstrate overall compliance of the TOE.

It is therefore suggested that the rationale justification for new functional components is limited to a list of the new elements, together with a statement that none of the new functional elements require new assurance techniques. Of course, if any of the new elements are so specialised that new techniques are required, these will have to be defined in the form of new assurance components, and suitable cross-references given.

For new assurance elements, an indication of how each new assurance element would be evaluated by evaluators should be given, equivalent to the information provided in the Common Evaluation Methodology for CC Part 3 elements.

There is much more information about selecting appropriate security requirements in earlier chapters of this Report.

12.7  TOE Summary Specification

The final section of a Security Target is the TOE Summary Specification. It provides a high-level definition of the security functions and assurance measures within the actual product or system that implement the security requirements. It is covered in a single criteria family, ASE_TSS. There are two developer action requirements:

ASE_TSS.1.1D  The developer shall provide a TOE summary specification as part of the ST.

ASE_TSS.1.2D  The developer shall provide the TOE summary specification rationale.

The content requirements are somewhat curious. Much of the information that would normally be provided in a rationale is asked for in the summary specification. Indeed, CC Trial Use Version 2.4 does away with the requirement for a rationale.
There are ten content requirements. The first asks for a description of the security functions and assurance measures within the actual TOE:

**ASE_TSS.1.1C**  *The TOE summary specification shall describe the IT security functions and the assurance measures of the TOE.*

This requirement could be interpreted as requiring much of the TOE’s functional specification, high level design and development methodology to be reproduced in the ST. This is clear duplication of TOE evaluation tasks, and this has been recognised in CC Trial Use Version 2.4, where there is a single TOE Summary Specification content requirement:

**V2.4_ASE_TSS.1.1C**  *The TOE summary specification shall describe how the TOE meets each SFR.*

It is therefore recommended that the TOE Summary Specification is kept as brief as possible. The objective should be to avoid duplication of information held in design documents that will be verified in TOE evaluation, and in particular not to pre-empt TOE evaluation tasks.

For each SFR, the mechanism that implements that SFR should be identified in the TSS, in terms that are consistent with the functional descriptions in the TOE’s functional specification and high level design. This need only be done in an informal style, even if the design documents are more formally structured:

**ASE_TSS.1.3C**  *The IT security functions shall be defined in an informal style to a level of detail necessary for understanding their intent.*

There is a specific requirement to identify where probabilistic or permutational mechanisms are used:

**ASE_TSS.1.9C**  *The TOE summary specification shall identify all IT security functions that are realised by a probabilistic or permutational mechanism, as appropriate.*

There are a number of specific content requirements concerning traceability that must be satisfied, and which would normally be regarded as rationale:

**ASE_TSS.1.2C**  *The TOE summary specification shall trace the IT security functions to the TOE security functional requirements such that it can be seen which IT security functions satisfy which TOE security functional requirements and that every IT security function contributes to the satisfaction of at least one TOE security functional requirement.*

**ASE_TSS.1.4C**  *All references to security mechanisms included in the ST shall be traced to the relevant security functions so that it can be seen which security mechanisms are used in the implementation of each function.*

Finally, where individual strength of function claims are made, these must be traced through:
ASE_TSS.1.10C  The TOE summary specification shall, for each IT security function for which it is appropriate, state the strength of function claim either as a specific metric, or as SOF-basic, SOF-medium or SOF-high.

The single content requirement for assurance measures concerns traceability:

ASE_TSS.1.7C  The TOE summary specification shall trace the assurance measures to the assurance requirements so that it can be seen which measures contribute to the satisfaction of which requirements.

It is therefore recommended that the assurance measures are described purely in terms of standards and procedures used by the developers. Each standard or procedure should be traced back to relevant SARs. If there are SARs which are not addressed by existing development standards, then it is likely that it will be difficult to demonstrate to the TOE evaluators that necessary development assurance tasks have been performed.

The solution to this problem is to produce the necessary standards and put them into use. This will solve both the ST evidence problem and the later and more serious problem of providing the necessary evidence for successful TOE evaluation.

There are three TOE Summary Specification rationale content requirements:

ASE_TSS.1.5C  The TOE summary specification rationale shall demonstrate that the IT security functions are suitable to meet the TOE security functional requirements.

ASE_TSS.1.6C  The TOE summary specification rationale shall demonstrate that the combination of the specified IT security functions work together so as to satisfy the TOE security functional requirements.

ASE_TSS.1.8C  The TOE summary specification rationale shall demonstrate that the assurance measures meet all assurance requirements of the TOE.

These should all have been satisfied within the summary specification already, in response to the traceability requirements. The rationale should therefore be reduced to cross-referencing the relevant evidence in the summary specification.

As already mentioned, there is no TOE Summary Specification rationale requirement in CC Trial Use Version 2.4.

Producing the TOE Summary Specification is dependent on the TOE design documentation, and beyond the scope of this Report. It is essential to avoid inconsistencies between the ST and TOE evidence, including TOE evidence that has not yet been produced at the time of writing the ST. The TOE Summary Specification should therefore be kept as short and general as possible, whilst remaining consistent with the specific content requirements above.
12.8 PP Differences

There are very few differences in CC Version 2.2 between the Protection Profile (APE) and Security Target (ASE) assurance classes, and even less in Trial Use Version 2.4.

It is unfortunate that the structure of the CC requires that the common information is presented twice. Worse, there is a real risk that intentional differences will be missed.

There are four differences between PPs and STs:

• There are minor changes to the introductory material since PPs have no TOE details;
• In Version 2.2, PPs cannot claim conformance to other PPs;
• PPs may contain incomplete operations in requirements;
• PPs have no TOE Summary Specification.

Otherwise the contents are identical and not discussed further in this section.

Each area of difference will now be described in turn.

Within the ST Introduction, ASE_INT.1.1C requires that the ST identification provides the labelling and descriptive information necessary to control and identify the ST and the TOE to which it refers. Protection Profiles are not TOE specific, and may need to be catalogued, so this requirement is modified:

APE_INT.1.1C The PP introduction shall contain a PP identification that provides the labelling and descriptive information necessary to identify, catalogue, register, and cross reference the PP.

This is a poorly worded requirement, as it is not defined what labelling and descriptive information is necessary. CC Trial Use Version 2.4 simplifies this requirement into just providing a unique method of identification:

V2.4_APE_INT.1.2C The PP reference shall uniquely identify the PP.

The name of the organisation developing the PP and an internal unique reference number should be sufficient to meet this requirement.

Also within the introductory material, ASE_DES.1.1C requires the TOE description to describe the product or system type, and the scope and boundaries of the TOE in general terms in both a physical and logical way. Since this is implementation specific, the equivalent PP requirement is worded in more general terms:

APE_DES.1.1C The TOE description shall describe the product type and the general IT features of the TOE.

CC Trial Use Version V2.4 breaks this requirement into three specific tasks:

V2.4_APE_INT.1.4C The TOE overview shall identify the TOE type.
V2.4_APE_INT.1.3C  The TOE overview shall summarise the usage and major security features of the TOE.

V2.4_APE_INT.1.5C  The TOE overview shall identify any non-TOE hardware/software/firmware available to the TOE.

In practice, there is no difference between the PP and ST requirements. Although the PP requirement asks for a less specific description, there is no TOE Summary Specification to expand on it. Thus the PP description will probably need about the same level of detail as would be put in an ST description.

Finally, there is no PP equivalent to ASE_INT.1.3C in CC Version 2.2. ASE_INT.1.3C requires a statement of any evaluable claim of CC conformance for the associated TOE. Since PPs are not TOE specific, there cannot be any such claims.

In Version 2.2, none of the PP Claims requirements apply to PPs, since PPs cannot claim conformance with other PPs. If the developer of a PP wants to include another PP within its requirements, that PP must be expanded in full and each section of text included within the equivalent section of the new PP. Note that its rationales will have to be modified to reflect the additional requirements of the new PP.

In CC Trial Use Version 2.4, this restriction does not apply and it is possible to claim conformance to other PPs and packages:

V2.4_APE_CCL.1.5C  The conformance claim shall identify all PPs and security requirement packages to which the PP claims conformance.

Note that rationales in the new PP will have to take account of the contents of the PPs and packages incorporated by reference. This limits the usefulness of this new facility.

Within the Security Requirements section, it is not necessary to complete all operations. The ST requirement to identify and perform all operations is replaced by two content requirements:

APE_REQ.1.6C  All completed operations on IT security requirements included in the PP shall be identified.

APE_REQ.1.7C  Any uncompleted operations on IT security requirements included in the PP shall be identified.

This is handled somewhat differently in CC Trial Version 2.4. There is no PP equivalent to V2.4_ASE_REQ.*.3C, which states that all assignment and selection operations must be completed, so the issue of completion is therefore left open.

Either way, this means that it is open to the PP developer to complete operations, but not mandated. PPs are also more likely than STs to have unsatisfied dependencies from security requirements, justified in the requirements rationale. Like operations that have to be completed, satisfaction of these dependencies may depend on TOE specifics that are not defined in the PP.
Finally, none of the TOE Summary Specification requirements apply to PPs as they must be independent of TOE implementation details. The whole section is omitted, in both Version 2.2 and Trial Use Version 2.4.
13 CONCLUSIONS AND COMMENTS

13.1 Introduction

In this chapter we draw conclusions, not only about this Report, but also concerning the Common Criteria and other guidance reports.

13.2 Conclusions About the Guidance Offered by this Report

13.2.1 The Methodology Works

It has been possible to generate guidance for constructing Protection Profiles and Security Targets based on the concepts of Security Problem Definition packages and Objectives Packages. The worked example in [1] successfully demonstrates the viability of the approach. Unlike previous guidance documents such as the PPST Guide [7], it has been possible to produce a methodology that describes how to perform each step in creating PPs and STs, not just what has to be done.

13.2.2 Use of Threat Analysis

The methodology for creating SPDs has been successfully based on threat analysis. Most aspects of policies and assumptions are explained in terms of threats assumed or ignored. The approach can handle both risk acceptance and risk avoidance.

13.2.3 Usability of JISEC Threats Database

The JISEC Threats Database and methodology for its use [11] have been shown to be useable in practice, and compatible with other aspects of PP and ST development.

13.2.4 Assumptions that Discount Threats

Assumptions that are created to discount threats do not form part of the CC methodology or requirements, but are very useful in showing completeness of coverage, and to enable customers to understand what the system or product does not do.

13.2.5 Objectives Interdependencies

Objectives Packages are very useful in breaking down the implementation of security requirements into manageable work units, but interdependencies between objectives that cross package boundaries add a degree of complexity and extra consistency checking.

13.3 Other Conclusions

13.3.1 Mandated Security Functionality

Sometimes security functionality is present in a PP or ST not to counter threats, but to offer security functionality for use by users and applications built upon the TOE.
Within the CC, the requirements to include this functionality have to be derived from an Organisational Security Policy. This is a viable but inelegant approach.

13.3.2 Assumptions and Environmental Objectives

Where assumptions about the operational environment are used to counter threats to the TOE, the CC requires that identically worded environmental objectives are generated that correspond to these assumptions, although these objectives are never validated or otherwise used. This is a pointless bookkeeping exercise, requiring redundant work. It would be possible to change the way that assumptions and environmental objectives are used to avoid this duplication.

13.3.3 CC Trial Use Version 2.4 Solves Many PP and ST Problems

The analysis of all CC Version 2.2 PP and ST Developer actions in Chapter 12 of this Report showed that there are very many problems of interpretation and duplication in CC Version 2.2 that have been identified and resolved in Trial Use Version 2.4. It is important that this good work is incorporated into the wider work on CC revision and not thrown away.

13.3.4 There Are Still Problems with the PP and ST Criteria

The preparation of this Report has exposed a number of problems in the CC criteria that have not been adequately addressed, either in the current CC Version 2.2, or in current proposals for changes such as [6]. For what appear to be good reasons, the guidance in this Report is also inconsistent with previous guidance documents such as the PPST Guide [7] and the US Consistency Instruction Manuals [9], [10].

13.3.5 Relationship Between the TOE and its IT Environment

There are many problems with the CC in expressing the relationship between a TOE and its IT operational environment. CC Version 2.2 is totally inadequate in this area, and Trial Use Version 2.4 gives little assistance in solving the problems. This Report has not been able to address most of the relevant issues in adequate detail. This may be a critical barrier in the future to building composite systems.

13.3.6 PP Compliance Claims

Claims of ST compliance with PPs cause problems because PPs cover both SPD and OP issues in one unit. It is not possible to use threat/policy/assumption compliance criteria and objective/SFR/SAR compliance criteria together without severely restricting the overall applicability of the combination. Current CC proposals to make PP compliance criteria more flexible do not really address this issue.
14 BIBLIOGRAPHY


APPENDIX A – EXAMPLE POLICIES

This Appendix contains some examples of policies, both well and badly expressed.

The main text explains that there are two uses for policies – policies that a particular control or type of control will be provided by the TOE, and policies that are used to require protection against threats or type of threats without further justification.

Here are examples of policies that mandate that particular security functions will be provided by the TOE:

P.TRACE The system will provide the ability to review the actions of both ordinary and privileged users.

P.IDAUTH Administrators will authenticate themselves before accessing any TOE functions or data.

Typically, we trust administrators, so why do need we bother to authenticate them? The policy above gives no reason, it just says we will do so.

P.MANAGE The TOE will provide authorised administrators with a set of administrative tools to manage effectively the TOE’s security functions and data.

Again, no reason, just do it.

P.EMERGENCY The TOE will at all times provide a capability to perform an emergency erase of all user data.

P.PRIVILEGE A record shall be kept of all persons to whom privilege is granted, and of the extent of such privileges.

It is not at all clear what the policy above might achieve, but the system can do it, so it is a legitimate policy.

P.ENCRYPT All confidential, proprietary, or otherwise sensitive information will be protected by encryption when transmitted between company buildings.

This is an example where the policy mandates a particular technique. We could, for example, have used physically secure communication links instead.

Here are some examples that mandate particular technologies:

P.CRYPTO Triple DES encryption will be used to perform all cryptographic operations.

P.ACCESS_BANNER The system will display an initial banner describing restrictions of use, legal agreements, and any other appropriate information to which users consent by accessing the system.
The policy above is an example of “best practice” not linked to any specific threat or threats.

**P.VULNERABILITY_TEST** The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.

Another good example – the requirement is not linked to any analysis of possible attackers or their capabilities.

**P.EAL** IT countermeasures will be assured to a level equal to EAL3 as a minimum.

A policy that sets the assurance objective, without any justification or explanation.

Policies are also used to require protection against threats, whether or not the threats actually exist. For example:

**P.INTEGRITY** The system will protect system data in transmission between distributed parts of the protected system.

There could be threats or not, but we will protect data anyway.

**P.MANAGE** The TOE will be managed and maintained so that its security functions are implemented and preserved throughout its operational lifetime.

This is a policy which covers certain types of unspecified threats relating to operation of the system. Note that this policy implies the TOE contains functionality preserving its security functions. If the controls were purely procedural and managerial, this policy should have been expressed as an assumption about the operational environment.

Here is an example of a policy that replaces the threats relating to uncontrolled distribution of sensitive information:

**P.NEED_TO_KNOW** The system will limit the access to, modification of, and destruction of the information in protected resources to those authorised users who have a “need to know” for that information.

And here are examples of policy decisions on how particular types of threat will be countered:

**P.ACCOUNTABILITY** The authorised users of the system will be held accountable for their actions within the system.

**P.CLEARANCE** The system will limit access to labelled data to authorized users whose clearance levels are appropriate for access to information bearing that label.

**P.SEPARATION** The TOE will only allow compartmented data to be transferred between users with the same Access Group ID.

These last two examples represent information flow policies.
Here are some examples of policies that are not well expressed. Of course, in particular circumstances, they might be legitimate policies when considered in combination with other policies and assumptions:

**P.INSTALL**  Appropriate action will be taken to investigate and resolve any insecure configuration warning messages generated during product installation.

This is actually an assumption about the environment – if the TOE could identify and fix the problem, presumably there would be no need to produce the warning.

**P.ALERT**  There shall be a prompt and appropriate response by the administrators in the event of a real-time warning being generated by the TOE.

This is an assumption about administrative procedures in the operational environment. It is hard to see how the TOE would enforce that “appropriate action” is taken.

**P.PRIVILEGE_REVOCATION**  When a privileged user or server-side administrator no longer requires a particular privilege, that privilege shall be withdrawn without delay from the user or administrator concerned.

How can the TOE know when a privilege is no longer required in order to withdraw it automatically?

**P.LOCATION**  The TOE shall be located within controlled access facilities or in an environment that prevents unauthorized physical access.

This should have been an assumption.
APPENDIX B – EXAMPLE ASSUMPTIONS

This Appendix contains some examples of assumptions, both well and badly expressed.

The main text explains that there are two forms of assumptions – assumptions that a particular control or type of control will be provided by the operational or development environment, and assumptions that a particular threat or type of threat does not exist.

Here are two examples of the first of these forms:

\textit{A.PHYSICAL} The TOE is located in a physically secure location.

\textit{A.PROTECT} The TOE hardware and software is protected from unauthorized physical modification.

These two assumptions represent assumptions about physical controls in the operational environment.

Here is an example of the second form:

\textit{A.ADMIN\_NO\_EVIL} Administrators are not careless, wilfully negligent or hostile, and can be trusted to perform all their duties in a competent manner.

This assumption does not represent a control or type of control; there is nothing in this statement that prevents administrators from behaving maliciously. Instead, it states that the threat that administrators misuse their ability to bypass controls on ordinary users – maliciously or otherwise – is discounted.

It is possible to recast the assumption into the first form:

\textit{A.EVIL\_ADMIN} If administrators are careless, wilfully negligent or hostile, their actions will be detected and handled by auditing or management controls outside of the TOE.

This is an equivalent to \textit{A.ADMIN\_NO\_EVIL} where we expect the environment to deal with the associated threats. But note that whereas \textit{A.ADMIN\_NO\_EVIL} trusted administrators completely, this form requires controls to be implemented in the operational environment to check that administrators are behaving properly.

Assumptions may work in collaboration with TOE controls. Here are two examples of that imply the TOE performs specific actions, such as generating audit logs or generating alarms:

\textit{A.AUDIT\_CHECK} Audit logs are frequently checked by administrators and any indications of improper activity investigated and action taken where required.

\textit{A.ALERT} If the TOE generates a real-time warning, administrators take prompt and appropriate action.
These two assumptions are used to handle cases where the TOE cannot prevent suspect activity, but can detect it.

Here is a further example of the second form:

A.BENIGN_USERS Authorised users are expected to act in a cooperative manner and not to deliberately disable or circumvent security controls.

This is the equivalent assumption to A.ADMIN_NO_EVIL for ordinary users in a TOE that requires user collaboration. Note that although this assumption presumes that there will be controls relevant to authorised users, it is not an assumption about the operation of those controls, it discounts the threat that users will bypass them.

Here are some examples of procedural assumptions:

A.CLEARANCE Procedures exist for authorising or refusing users access to specific security levels or compartments.

A.SENSITIVITY Procedures exist for establishing the security level and compartmentalisation of all information imported into the system.

A.TOE_CONFIG The TOE is installed, configured, and managed in accordance with its evaluated configuration.

This last assumption addresses threats of disabling TOE controls through incorrect configuration. It could be better worded – either to make it clear that following defined installation and operating procedures is necessary in order to create and maintain the evaluated configuration, or perhaps to indicate that the TOE is so easy to configure and operate that the threats of incorrect configuration can be ignored.

Here are some examples relating to technical functionality outside the TOE:

A.COMMUNICATIONS The communications infrastructure provides communications that are reliable and resilient against failure.

A.NETWORK Network services supporting the distributed TOE are based on secure communications protocols which ensure the authenticity of remote users.

Both the above concern communications threats, and assert that they are countered by the environment.

Here are some other examples of assumptions about controls in the operational environment that would be implemented by IT:

A.TIME_SOURCE The IT environment will provide a time service that can be used by the TOE to reliably represent the date and time of day.

A.ACCESS_CONTROL The operating system upon which the TOE software runs will be configured to restrict modification to TOE executables and configuration files to only Authorised Administrators.
Note that this assumption for this TOE would become a policy for the operating system’s Security Target, if it had one.

A.BACKUP  Backups of all information held by the TOE including configuration parameters are sufficiently detailed and sufficiently current to restore TOE operation in the event of a failure or security compromise.

This is an interesting assumption – it implies that backup capabilities will exist, but are not part of the TOE.

The next example is particularly complex:

A.SIGNED_CODE  Malicious code destined for the TOE is not signed by a trusted entity.

This assumption discounts the threat that malicious code will be signed by a trusted source. It implies that other types of malicious code will be rejected by other means. However, although it deals with the operation of technical controls, it is used to discount a particular type of threat. Therefore it should be an assumption.

Here is an example of a complex physical security assumption:

A.CONSOLE  A physically secure console exists that can be used by administrators to perform management functions regardless of ordinary user activity.

Note that this assumption is countering a certain type of threat – namely that ordinary user activity could lock out the administrators. However, it is specifying how it is done by requiring a physical control. If the administration console was located in a general area, where it can be damaged or disabled by users (even accidentally), the assumption would not hold.

Assumptions can be very general in form:

A.NOBYPASS  Protected resources cannot be accessed in a way that bypasses the TOE.

This is an interesting assumption. Some STs may imply it by requiring the presence of physical controls.

A.ENVIRONMENT  The following threats are countered by the TOE environment: ...

This is an interesting approach to discounting certain types of threat. It implies that certain threats do exist, but places the problem of eliminating them on the designer of the TOE environment. This may limit the usefulness of the TOE, but also it does not constrain the design of the operational environment.

Finally, here are some examples of assumptions that are not well expressed. Of course, in particular circumstances, they may be legitimate assumptions when considered in combination with other assumptions and TOE policies:
A.ATTACK_POTENTIAL  The attack potential of malicious attacks aimed at discovering exploitable vulnerabilities is moderate.

This implies either that more sophisticated attacks will be countered by the environment or that they may be discounted. It would be better expressed as:

A.ATTACK_POTENTIAL  The attack potential of malicious attacks aimed at discovering exploitable vulnerabilities is no higher than moderate.

This tells us clearly that more sophisticated attacks can be ignored.

Alternatively, this assumption could be recast as a requirement:

P.ATTACK_POTENTIAL  The TOE will protect against malicious attacks aimed at discovering exploitable vulnerabilities with an attack potential of up to moderate.

This is expressed as a policy since it will be implemented in the TOE.

Here is another assumption that should really be a policy:

A.DEV_TOOLS  There are no applications development capabilities that could be used to create and execute arbitrary code within the TOE.

This could be implemented as an operational environment control (see A.ACCESS_CONTROL above), but as worded it could also be a characteristic of the TOE. It should therefore be reworded as a policy:

P.DEV_TOOLS  The system will not contain applications development capabilities that could be used to create and execute arbitrary code within the TOE.

Written as a policy, it could then be implemented by either technical or procedural means.

Here is a final badly worded assumption:

A.DISASTER  The TOE and its environment have sufficient protections and controls in place to protect the availability of the TOE from natural disasters such as fire or flood, as well as catastrophic failure of power supply and communications.

In one sense, this is a good assumption – natural disasters are often discounted as threats without being mentioned explicitly, so the inclusion of this assumption as an exclusion is good. However, it is badly worded because it mentions the TOE as providing protection. If the TOE provides controls, then they should be included in the ST and traced back to natural threats. If the TOE provides no protection, the following wording would be better for this assumption:

A.DISASTER  The environment surrounding the TOE has sufficient protections and controls in place to protect the availability of the TOE from natural disasters such as fire or flood, as well as catastrophic failure of power supply and communications.
This Appendix contains some examples of security objectives, both well and badly expressed.

There are two basic types of security objectives – TOE objectives and environmental objectives.

Here are some examples of TOE objectives, starting with a general objective for user binding:

**O.AUTHORISATION** The TOE must ensure that only authorised users gain access to the TOE and its resources.

Here are some secondary objectives for user binding that could be needed as a result of specific threats or policies (perhaps the threat of unauthorised access following successful impersonation, and a policy to meet legal requirements to warn of unauthorised use):

**O.ACCESS_HISTORY** The TOE must display information to authorised users on authentication relating to previous successful and failed authentication attempts.

**O.BANNER** The system must display an advisory warning regarding legal use of the system.

Here is a classic objective for TOE security behaviour:

**O.DISCRETIONARY_ACCESS** The TOE must control access to resources based on identity of users. The TOE must allow authorized users to specify which resources may be accessed by which users.

And here is an example of a more specific form:

**O.SEPARATION** The TOE must provide a security domain for its own execution that protects it from modification by anyone other than authorised administrators.

Here is a very specific communication objective, that presumably exactly matches an external policy decision. It applies to all uses of the RSA algorithm, which must have been selected by some other policy.

**O.RSA_KEYGEN** The TOE will generate RSA key pairs with 1024 bit key length.

Here is a general audit objective. It has two separate aspects, and could have been split into two objectives – but presumably both requirements address the same threats, so there is no point separating the two aspects from each other:

**O.AUDITING** The TOE must record the security relevant actions of users. The TOE must make this information accessible to authorised administrators.
Here is an example of a supporting objective to ensure that the TOE cannot be impersonated by another entity (for example to counter the JISEC Database threat T.PRETEND_TOE). It could be regarded as either a communication or user binding objective. Indeed, CC Version 2.2 treats Trusted Path as a separate functional class.

**O.TRUSTED_PATH** The TOE must provide the capability to allow a user to ensure that they are not communicating with some other entity pretending to be the TOE during initial user authentication.

And here is a TOE objective for resource management:

**O.ADMIN** The TOE must provide functions to enable system administrators to effectively manage and maintain the TOE and its security functions, ensuring that only they can access administrative functionality.

This is expressed in very general terms. If precise information on threats (for example, from a full risk analysis) was available, it might look more like this:

**O.ADMIN** The TOE shall limit TOE administrative functions to those verified as administrators by the TOE. The TOE shall provide all the administrative functions necessary to support the management of TOE security and shall include functions to: 1) tune the performance of the system to meet the requirements of the availability policy; 2) maintain the security attributes associated with users; 3) manage auditing (including accessing or modifying the audit trail); 4) restore the system to a secure state in the event of failure or interruption; 5) verify secure operation of the TOE.

Now here are examples of environmental objectives that are physical or procedural. Note that the names of these objectives start with “OE” to indicate that they cannot be implemented by the TOE:

**OE.PHYSICAL** The TOE must be protected from physical attack.

**OE.ADMIN** Administrators must be provided with the supporting facilities necessary to perform their allotted functions.

The two objectives above are very general. It may be helpful to those responsible for operating the TOE to be more precise in the requirements, for example:

**OE.PHYSICAL** Those responsible for the TOE environment must ensure that those parts of the TOE that enforce security policy are protected from physical attack which might compromise TOE security objectives.

**OE.AUDSUP** The environment must contain the tools necessary to effectively review the audit trail.

For the final example in this group, here is an assumption about installation of the TOE:

**A.INSTALL** Those responsible for the TOE environment will ensure that the TOE is delivered, installed, managed, and operated in a manner which does not compromise the TOE security objectives.
This can be echoed as the objective, using identical wording.

OE.INSTALL Those responsible for the TOE environment will ensure that the TOE is delivered, installed, managed, and operated in a manner which does not compromise the TOE security objectives.

Here are examples where the environment must provide supporting IT security controls:

OE.OSAUTH The host operating system must identify and authenticate users prior to providing access to any TOE facilities.

OE.OSFAIL The host operating system and database must provide a means for secure recovery from failures.

The main text explains that the CC currently does not handle development environment objectives very well. Some have to be treated as TOE objectives as they will generate SARs. Others will not generate SARs and so have to be treated as environmental objectives. Here are examples of development objectives that generate SARs:

O.DESIGN The TOE will be built using semiformal design principles and techniques which are accurately documented.

O.VULNERABILITY_TEST The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.

Both the above can be expressed as SARs and so checked by evaluation. However, the following objective is an exclusion and so has to be treated as an environmental objective:

OE.TROJAN The developers of the TOE will be trusted not to insert covert functionality within the TOE that could be used to bypass security controls.

Finally, here are some examples of objectives that are not well expressed. Of course, in particular circumstances, they may be legitimate objectives, for example if they reflect assumptions or policies.

The first example is of object reuse, part of resource management:

O.RESIDUAL_INFORMATION The TOE must ensure that any information contained in a protected resource is not released when the resource is recycled.

The problem with this objective is that it is very badly worded. What does “not release” mean? The normal implementation of object reuse is to ensure that any previous contents are overwritten or destroyed. “Not released” could be interpreted as meaning that the information is passed on to the next user!

O.TAMPER The TOE must prevent physical tampering with its security critical parts.
This is also very badly worded. It is highly unlikely that the TOE can prevent physical tampering – unless, perhaps, it is supposed to electrocute anyone who touches it! It is more likely that the intended objective is for it to destroy its internal data and processing capability if it detects an attempt at tampering.

O.MANAGE The TOE must incorporate user friendly mechanisms to ensure secure administration of its operation.

The problem with the objective above is that mechanisms can be user friendly, but that does not mean that they will ensure secure operation! It should probably have been worded as:

O.MANAGE The security administration functions provided by the TOE must be user friendly in operation.

This would be a perfectly reasonable requirement to counter the threat that administrators might make accidental mistakes that breach security.

O.FLAW The TOE must not contain flaws in design, implementation or operation.

This is a development environment objective. Unfortunately in the real world it is unprovable! At a practical level, it cannot be used to derive SARs for evaluation (all the relevant SARs search for flaws, they do not guarantee that none remain); but equally it is not worded as an assumption, to be taken as true without further investigation.