

# **EMV®**

## **Level 3 (L3) Testing Framework**

---

### **Implementation Guidelines**

Version V1.1  
April 2019

## Legal Notice

This document summarizes EMVCo's present plans for evaluation services and related policies and is subject to change by EMVCo at any time. This document does not create any binding obligations upon EMVCo or any third party regarding the subject matter of this document, which obligations will exist, if at all, only to the extent set forth in separate written agreements executed by EMVCo or such third parties. In the absence of such a written agreement, no product provider, test laboratory or any other third party should rely on this document, and EMVCo shall not be liable for any such reliance.

No product provider, test laboratory or other third party may refer to a product, service or facility as EMVCo approved, in form or in substance, nor otherwise state or imply that EMVCo (or any agent of EMVCo) has in whole or part approved a product provider, test laboratory or other third party or its products, services, or facilities, except to the extent and subject to the terms, conditions and restrictions expressly set forth in a written agreement with EMVCo, or in an approval letter, compliance certificate or similar document issued by EMVCo. All other references to EMVCo approval are strictly prohibited by EMVCo.

Under no circumstances should EMVCo approvals, when granted, be construed to imply any endorsement or warranty regarding the security, functionality, quality, or performance of any particular product or service, and no party shall state or imply anything to the contrary.

EMVCo specifically disclaims any and all representations and warranties with respect to products that have received evaluations or approvals, and to the evaluation process generally, including, without limitation, any implied warranties of merchantability, fitness for purpose or non-infringement. All warranties, rights and remedies relating to products and services that have undergone evaluation by EMVCo are provided solely by the parties selling or otherwise providing such products or services, and not by EMVCo, and EMVCo will have no liability whatsoever in connection with such products and services.

## Revision Log – Version 1.1

Version	Date	Description
V1.0	March 2017	First public release of document.
V1.1	April 2019	<ul style="list-style-type: none"><li>Include explicit requirements for TSE Test Tool Vendors in <b>Section 3.2: TSE Requirements</b>.</li><li><b>Section 4.1.1: TSE Test Set Files Format</b> indicates that “the headers can appear in any order”.</li><li><b>Section 4.1.3: Question Definition File Format</b> – introduction of an “Attributes” field.</li><li><b>Section 4.2.1: TestRunInfo.xml</b> – Inserted &lt;PSI&gt; as part of the TrackingNo’s unique identifier field.</li><li><b>Table 4.14: Online message format specification</b> – updated the Description of the FieldBinary field.</li><li>Other editorial updates throughout.</li></ul>

# Contents

<b>1</b>	<b>Background .....</b>	<b>8</b>
1.1	Technical L3 Components .....	8
1.2	Audiences .....	9
1.3	Objectives .....	9
1.4	Document Organization .....	10
<b>2</b>	<b>L3 Test Environment Architecture .....</b>	<b>11</b>
<b>3</b>	<b>L3 TSE Requirements .....</b>	<b>14</b>
3.1	Definitions .....	14
3.2	TSE Requirements .....	16
<b>4</b>	<b>EMVCo L3 Detailed Formats.....</b>	<b>23</b>
4.1	TSE Test Set Files.....	23
4.1.1	Test Set File Considerations .....	23
4.1.2	Test Selection File Format .....	29
4.1.3	Question Definition File Format.....	30
4.1.4	Error Definition File Format.....	32
4.1.5	Suggestion Definition File Format .....	33
4.1.6	Information Report File Format .....	34
4.1.7	Card File Format.....	35
4.1.8	Test Case File Format .....	36
4.1.9	Pass Criteria File Format .....	37
4.1.10	Test Reference File Format .....	38
4.1.11	Manifest File Format .....	39
4.2	TSE Test Session Files .....	40
4.2.1	TestRunInfo.xml.....	41
4.2.2	RuleSet.xml .....	43
4.2.3	TestRun.xml.....	47
4.2.4	Selected.xml .....	48
4.2.5	Manifest file.....	49
4.3	Tool Pass/Fail Automation Criteria .....	50
4.4	Test Report .....	58
4.5	Machine-readable Test Card Image File Format.....	59
4.5.1	EMVCo Level 3 Card Application Behavior .....	60
4.5.2	EMVCo Level 3 Card Image Definition.....	64
4.6	Card Terminal Log Format.....	65
4.6.1	Context .....	65
4.6.2	Overview.....	65
4.6.3	Log Details.....	66

4.6.4	Interface Logs .....	68
4.6.5	Signature .....	71
4.6.6	Schema and Validation .....	72
4.6.7	Examples .....	74
4.7	Common Online Message Format.....	80
4.7.1	Introduction .....	81
4.7.2	Format specification.....	81
4.8	TSE Validation Report Files.....	94
4.8.1	TSE Validation Report structure.....	94
<b>Annex A</b>	<b>Common L3 Terminologies .....</b>	<b>96</b>

## Figures

Figure 2-1: Test Environment Architecture .....	12
Figure 2-2: L3 TT and L3 CS Environment Architecture .....	13

## Tables

Table 4.1: Character substitution .....	27
Table 4.2: XML character substitution .....	45
Table 4.3: List of Syntax Fields and Definitions .....	51
Table 4.4: Basic Log .....	66
Table 4.5: Root Level Schema Fragment.....	66
Table 4.6: Log Details Example .....	67
Table 4.7: Log Details Schema Fragment .....	67
Table 4.8: Example Card Log Section.....	70
Table 4.9: Interface Log Schema Fragment .....	71
Table 4.10: Complete Log Schema .....	72
Table 4.11: Simple Log – illustrating use of signature .....	74
Table 4.12: Contact Online Transaction .....	75
Table 4.13: Contactless Online Transaction.....	78
Table 4.14: Online message format specification .....	83
Table 4.15: Online message sample .....	90

# 1 Background

The Terminal Integration Task Force (TITF) was established by EMVCo in September 2013, to examine the Payment Systems' testing processes for the integration of EMV contact and contactless acceptance devices into their payment environments. The task force's mission was to determine the feasibility of standardizing key elements of these individual processes, thereby delivering a more streamlined integration testing experience to Chip acquirers and acquirer processors (Testers) that support multiple Payment Systems.

The charter defined by EMVCo for the task force required a detailed review of the various processes that Testers encounter to meet individual Payment Systems' requirements for a terminal that has been approved by EMVCo and is ready for deployment in the field. The task was to determine the possibilities of creating standardized, consolidated processes, leveraging the EMVCo infrastructure as necessary. If opportunities were identified, the task force would present a proposal to the EMVCo Board for approval to implement as necessary.

The task force's initial efforts concluded that many elements of the individual processes could indeed be standardized for better efficiencies. The *EMVCo Level 3 Testing Framework – Process Enhancements (L3 Framework)* document that was originally published by the TITF in November 2014 and re-released in November 2016:

- Outlined the initial findings on synergies and differences across the various processes.
- Described the intended standardization areas, as identified by the participating Payment Systems that are members of EMVCo.
- Provided details of the next steps to be taken by the TITF to achieve a more standardized L3 testing process.
- Provided a timeline estimate on stages related to delivery of the Phase II initiatives.

**Note:** In January 2017, the EMVCo Board of Managers officially approved the renaming of the **Terminal Integration Task Force (TITF)** to the **Level 3 Testing Group (L3TG)**.

This document, the *EMVCo Level 3 Testing Framework – Implementation Guidelines (FIG)* is intended to be a companion document to the *L3 Framework*. It provides its targeted audiences with specific implementation details and instructions for each technical component of the *L3 Framework*.

## 1.1 Technical L3 Components

Through the recent work effort of the TITF, EMVCo has defined a set of standardized L3 Test Tool technical components, aimed at streamlining L3 testing efforts for key stakeholders. These enable:

- **For Test Tool Vendors:** a streamlined process for L3 Test Tool development and subsequent qualification.
- **For Users and Payment Systems:** improved automation, test execution, results submission and validation efforts during L3 testing.

These components include specified EMVCo L3 machine-readable formats for the following:

- A set of files (TSE Test Set files), covering instructions on how to collect terminal configuration information and build test plans, to streamline test plan generation activities.
- A syntax for expressing test case pass/fail result criteria. That syntax is used in the relevant TSE Test Set files.
- A test session file format (TSE Test Session file), covering terminal configuration data, a test plan and test report, to streamline test plan processing and validation activities.
- A test card image syntax, allowing representation of the expected behaviours for simulated test cards.
- Formats for both card-to-terminal and authorization message logs, to streamline log parsing by making it equipment neutral.

For each of the above components, standardized, machine-readable formats have been defined by EMVCo. These formats are the Extensible Mark-up Language (XML) and Comma Separated Value (CSV), chosen largely due to their openness and wide adoption across the industry.

## 1.2 Audiences

This document and its guidelines are intended primarily for use by L3 Test Tool vendors, Payment Systems, their Financial Institution clients and other L3 service providers and stakeholders.

## 1.3 Objectives

This document aims at providing the EMVCo L3 stakeholders the appropriate information and directives to implement the EMVCo L3 machine-readable files.

## 1.4 Document Organization

This document is organized as follows:

**Chapter 1: Background** – introduces this document (*EMVCo Level 3 Testing Framework – Implementation Guidelines*), describing the technical components it addresses, its intended audiences, its objectives and organization.

**Chapter 2: Level 3 Test Environment Architecture** – describes the three key components of the L3 test environment; the L3 Test Selection Engine (L3 TSE), the L3 Test Tool (L3 TT) Engine and the L3 Card Simulator (L3 CS).

**Chapter 3: EMVCo L3 TSE Requirements** – describes in detail the requirements for the L3 TSE Tool.

**Chapter 4: EMVCo L3 Detailed Formats** – describes in detail the machine-readable card image format, the Tools Pass/Failed Automation criteria, the Card Terminal log format, the common On-line Message format, the Test Set files and the Test Session files format.

**Annex A: Common L3 Terminologies** – includes a list of commonly used terms within the L3 testing environment.

## 2 L3 Test Environment Architecture

The L3 test environment architecture proposed by EMVCo consists of three key components:

- The L3 Test Selection Engine (L3 TSE)
- The L3 Test Tool (L3 TT) Engine
- The L3 Card Simulator (L3 CS)

The components have the following objectives:

**L3 TSE:** Third-party vendor-provided Test Selection Engine is intended to provide Clients (that often support multiple Payment Systems) with a convenient means of preparing applicable Test Session files for individual Payment Systems. The Test Session files are subsequently presented in a machine-readable format that will be compatible with any EMVCo-qualified L3 Test Tool. EMVCo will qualify the L3 TSE's capability to:

- Import the machine-readable TSE Test Set files provided as a TSEC-package by the Payment Systems. There will be one or more TSEC-packages per Payment System. It is a zip file with Test Set files which include instructions on how to collect terminal configuration information, test cases, test case applicability conditions, and pass criteria definitions. This can happen for multiple Payment Systems,
- Process the configuration provided by the TSE Test Set files by collecting the answers to the applicable set of questions presented to the user. Answering these questions is required in order to evaluate the L3 testing scope. If the client's terminal supports multiple Payment Systems, then this process needs to happen for each applicable TSEC-package separately.
- Extract the applicable test cases and related pass criteria.
- Build the corresponding individual TSE Test Session files that describe the applicable test plan for the terminal under test.
- Export the individual Test Session files grouped in a TSE-package (one per TSEC-package). This is a zipped folder with renamed extension .tse ready to be used by a L3TT tool.

**L3 TT:** Third-party vendor-provided test tool, qualified by EMVCo for the purpose of executing the selected Level 3 Test Cases required by clients or their service providers. The tool will be qualified for its technical capability to correctly:

- Import individual machine-readable Test Session files generated by the L3 TSE
- Execute selected Test Cases, correctly keeping track of the logs for them and determining the pass criteria verdicts.
- Import Card-to-Terminal logs in EMVCo L3 format
- Import authorization message logs in EMVCo L3 format
- Update individual Test Session files with the test results, and export the updated files in a TSEZ-package (a renamed zip file)

- Export Card-to-Terminal logs in EMVCo L3 format (as generated by the L3 Card Simulator, so L3TT is acting here as a pass-through agent)

**L3 CS:** Third-party vendor-provided test tool, qualified by EMVCo for the purpose of simulating the personalization images and behaviors of physical, non-programmable test cards. The tool will be qualified for its technical capability to correctly:

- Import Payment System-provided, machine-readable Test Card Images (see section 4.5 for a definition)
- Select the applicable Card Image as indicated by the user or the L3TT (based on the Test Case), and provide the appropriate responses to the terminal based on its Request commands
- Export Card to Terminal Logs in EMVCo L3 format

Figures 2.1 and 2.2 below provide illustrations of the interaction between these three components. It is allowed to combine components in one application, as long as the necessary interfaces to communicate with other vendor's components are available.

**Figure 2-1: Test Environment Architecture**

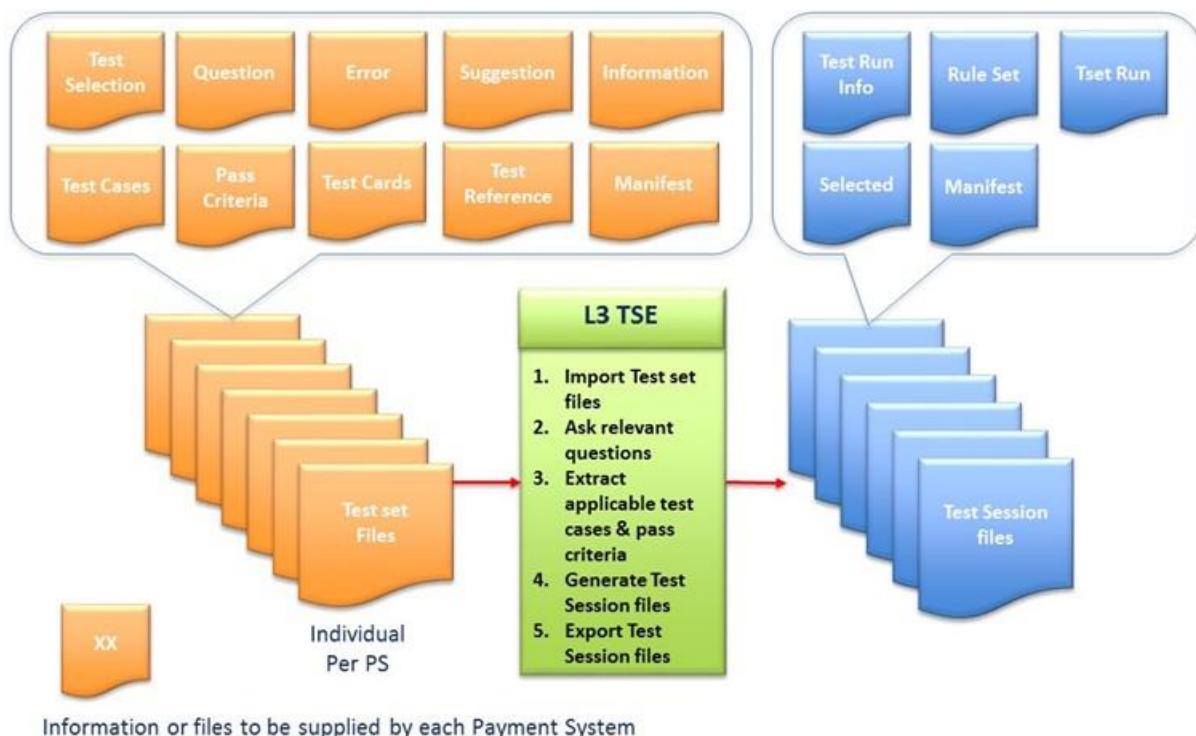
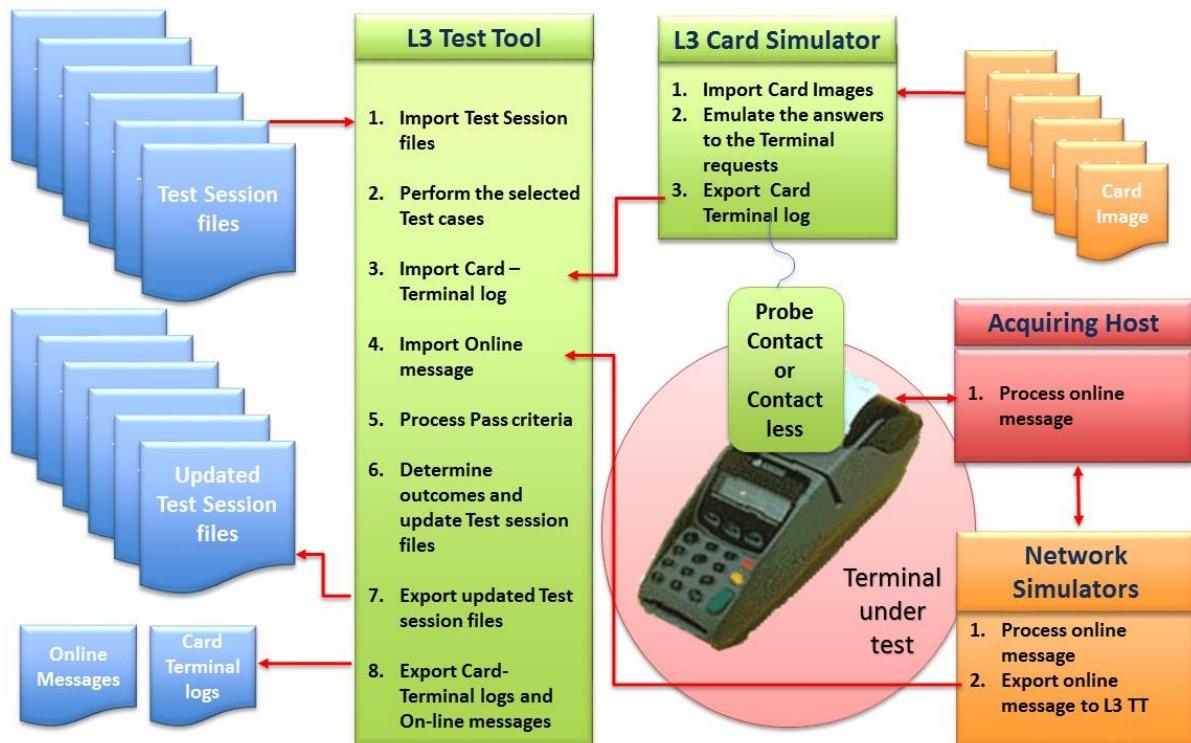


Figure 2-2: L3 TT and L3 CS Environment Architecture



# 3 L3 TSE Requirements

## 3.1 Definitions

### Terminal Integration Context

Describes the context required to create a test plan for all applicable Payment Systems for the specific terminal in its specific context.

It is a set of context statements. It describes both the terminal configuration as such (e.g. ODA support), and the environment in which it is operating (e.g characteristics of the Payment Network it is connected to: single / dual message).

### Context statement

The statement that a certain context expression is TRUE or FALSE.

### Context expression

An expression of the form *<criteria name> <operator> <value>*.

*criteria name*: Each Payment System might use their own criteria names. The Payment Systems shall ensure that their criteria names can be evaluated by asking dedicated questions in the Question Definition file. Next to that there is one special criteria name: date, which should have the value of the current date.

*operator*: The operator is one of the following: "=", "<>", ">", "<", "in", "==".

The "in" operator is used with group definitions to indicate that the context expression bit should be set if the answer provided is part of a group. For example, the applicable test cases might change on a regional basis. Rather than allocating a dedicated context expression bit to each country a single bit is allocated to each region. The Groups field in the Question definition file (see section 4.1.3) provides the list of values that are part of each group.

The "==" operator is used to compare the answers to list or set criteria questions i.e. the operator is comparing two answers rather than an answer with a fixed value.

*value*: Depending on the field type, the value part will either be a numeric value, a date in the format YYYYMMDD or a text string. In the case of criteria in numeric format the value may also be another criteria in numeric format.

Examples:

- CL\_CVM\_Limit = 0
- CL\_Transaction\_Limit = CL\_CVM\_Limit
- Term\_Type = Attended POS
- Term\_country in na (na being defined elsewhere as the North American countries)
- date > 20171115, (where date is the current date as defined above)

### Question

A user question to determine the value of a criteria name in the Terminal Integration Context.

The answer to a question might lead to multiple context statements. E.g. if the answer to `Term_type` is “Attended POS”, then `Term_type = Attended POS` is TRUE, `Term_type = ATM` is FALSE.

### **Applicability condition**

A logical expression of context statements that should be TRUE for a certain test case to be applicable, or for a question or error to be presented to the user.

If a context expression cannot be evaluated because it depends upon a yet unasked question, then

- \* for questions: it should be removed from the logical expression.<sup>1</sup>

- \* for errors: the error should be considered as not applicable.

### **Common question**

A question that shares the same criteria name and same question definition, with another question and featuring an attribute called ‘Common’ in the Question Definition file.

### **Timestamp**

Always in ISO-8601 format unless specified differently.

---

<sup>1</sup> Ignoring is different than putting the context statement to FALSE. Example: if `a=1 && b=1` is the expression and `b` is not yet evaluated, then the question is applicable when `a=1`. If `a=1 && not b=1` is the expression, still the question is applicable when `a=1`.

## 3.2 TSE Requirements

Note: if you choose to implement an optional feature, it shall be implemented in accordance with the requirement defined here.

Req	Requirement	M/O/C
1.0	L3TSE shall be able to import the TSE Test Set files (TSEC-Package, TSEC for short). This may be a developer/administrative function not available to end users.	M
1.1	<ul style="list-style-type: none"> <li>• L3TSE shall be able to import the csv files with headers presented in any order and ignore any unknown csv header.</li> </ul>	M
2.0	L3TSE shall only allow import of a TSEC that has a correct manifest file and shall check that	M
2.1	<ul style="list-style-type: none"> <li>• There are no files in the Test Set files that are not in the manifest file,</li> </ul>	M
2.2	<ul style="list-style-type: none"> <li>• All files listed in the manifest file shall be present,</li> </ul>	M
2.3	<ul style="list-style-type: none"> <li>• For each file the hash shall match,</li> </ul>	M
2.4	<ul style="list-style-type: none"> <li>• The manifest file shall have the correct signature.</li> </ul>	M
3.0	L3TSE shall have a user interface that presents the tester with the relevant questions, to determine the Terminal Integration Context.	M
4.0	While processing the TSE Test Set Files, the user interface shall process the character substitutions as shown in Table 4.1.	M
5.0	<p>L3TSE shall either process the HTML codes available in the Test Set files, or ignore any HTML code other than the line break (HTML code &lt;br&gt;).</p> <p>It is not acceptable when the HTML codes (like &lt;b&gt;) are shown to the user.</p>	M
6.0	To determine the Terminal Integration Context, L3TSE shall process the questions group by group, as indicated in the Question Definition file. Before presenting a group and dynamically after having received an answer to any of the group's questions:	M
6.1	<ul style="list-style-type: none"> <li>• L3TSE shall evaluate the Question applicability for each of the questions to understand whether the question shall be presented to the user. <ul style="list-style-type: none"> <li>- A question with attribute "Common" shall only be asked once, even when it appears multiple times in the Test Set files.</li> </ul> </li> </ul>	M
6.2	<ul style="list-style-type: none"> <li>• L3TSE shall check the Suggestion file, to understand whether the questions in the group come with applicable suggestions.</li> </ul>	M

	<ul style="list-style-type: none"> <li>- Suggestions shall be shown to the user.</li> <li>- It shall be possible for the user to override suggestions.</li> <li>- Questions that have a forced answer, shall not be shown to the user. Note that, if relevant, these questions will be visible in the information report as described below.</li> <li>- Answers that are highlighted to be removed shall no longer be visible to the user.</li> <li>- In case multiple rules apply follow the guidance in section 4.1.5.</li> <li>- If a user decides to step back through the questions and make a different selection then any forced, suggested or removed options that were based on that previous response need to be undone. Note that a suggested response to an already answered question does not change the answer! Any change to previously answered questions shall cause the tool to reevaluate the terminal integration context (defined previously).</li> </ul>	
6.3	<ul style="list-style-type: none"> <li>• The group of questions shall be presented in the following way: <ul style="list-style-type: none"> <li>- L3TSE shall present the questions in the order they appear in the Question definition file.</li> </ul> </li> </ul>	M
6.4	<ul style="list-style-type: none"> <li>• Questions that are grouped shall be presented together under the group prompt as specified in the Question definition file.</li> </ul>	M
6.5	<ul style="list-style-type: none"> <li>• L3TSE shall present help text available at group level and at question level to the user.</li> </ul>	M
6.6	<ul style="list-style-type: none"> <li>• It is allowed for L3TSE to suggest an answer to the user that does not come from the suggested answers from TSEC, but is suggested because the L3TSE tool knows the user. E.g. the Acquirer's Name might be prepopulated.</li> </ul>	O
6.7	<ul style="list-style-type: none"> <li>• The ways in which the user can answer to questions shall respect the different answer types indicated in the Question definition: <ul style="list-style-type: none"> <li>- A varBoolean shall only allow a Boolean choice.</li> <li>- A varNumber shall only allow a Numerical value.</li> <li>- A varList shall only allow one answer to be selected.</li> <li>- A varSet shall allow multiple answers to be selected.</li> <li>- An optional question (indicated by the mode field) shall not force an error if left unanswered.</li> </ul> </li> </ul>	M
6.8	<ul style="list-style-type: none"> <li>• L3TSE can use the guidance given in the mode field indicated in the Question definition.</li> </ul>	M

	<ul style="list-style-type: none"> <li>- A drop_down should be represented by a drop box.</li> <li>- For a multi_select two lists should be presented, a list of unselected items and a list of selected items with a mechanism to move items between the two lists.</li> <li>- If the mode explicitly mentions the number of lines / element (:N) this should be respected.</li> </ul>	
6.9	<ul style="list-style-type: none"> <li>• After having received the answers to the questions in the group L3TSE shall: <ul style="list-style-type: none"> <li>- Check whether the answer to the question has the correct type (E.g. Boolean, number, string, email address<sup>2</sup>)</li> </ul> </li> </ul>	M
6.10	<ul style="list-style-type: none"> <li>- Update the Terminal Integration Context</li> </ul>	O
6.11	<ul style="list-style-type: none"> <li>- Process the full Error file and display errors or warning messages as defined. <ul style="list-style-type: none"> <li>- If there is an error, the user interface shall move back to the question indicated by the "GotoQuestion" field (although the user must still be able to navigate to different questions on earlier pages).</li> <li>- If there is a warning, the user shall have the option of going back to the problem question indicated by the "GotoQuestion" field or continuing without seeing the warning again.</li> </ul> </li> </ul>	M
7.0	Finally, L3TSE shall <b>present the Information Report</b> for its Terminal Integration Context to the user, defined in section 4.1.6, as soon as all questions have been processed.	M
7.1	<ul style="list-style-type: none"> <li>• It shall present the headers at the right level, indicated by the HeadingLevel, followed by the indicated content from the Information Report file.</li> </ul>	M
7.2	<ul style="list-style-type: none"> <li>• All placeholders (see section 4.1.6) should be filled with the correct values.</li> </ul>	M
7.3	<ul style="list-style-type: none"> <li>• At this point, the presentation of the Information Report, L3TSE shall always give the user the opportunity to review the answer to the questions and to return to one of them to change it as documented below in the Question Review Process.</li> </ul>	M
8.0	L3TSE shall allow the <b>Question Review Process</b> :	M
8.1	<ul style="list-style-type: none"> <li>• When the user updates the answer to a question that has attribute "regression", L3TSE shall flag that this has happened by setting the MODE_Regression indicator.</li> </ul>	M

<sup>2</sup> Indicated by the mode in the question definition file, see section Question Definition File Format4.1.3

8.2	<ul style="list-style-type: none"> <li>When the user updates the answer to a question, L3TSE shall proceed from that point in the question file to the end to decide whether any succeeding questions need a new or a different answer. Reason for this requirement is that changing an answer to a question might change the applicability of other questions or restrict the allowed answers of them.</li> </ul> <p>Note that it is up to L3TSE to present all intermediate questions with the already given answers, or hide them for the user and only show the questions that needs revision.</p>	M
8.3	<ul style="list-style-type: none"> <li>When all questions are answered, L3TSE shall <i>present the Information Report</i> for the updated Terminal Integration Context as defined above.</li> </ul>	M
9.0	The user shall be able to view the Test Reference file (HTML), as defined in section 4.1.10.	M
10.0	If all questions are answered, L3TSE shall allow the user to export one package of TSE Test Session files per TSEC. The Test Session files shall adhere to the requirements given in section 4.2.	M
11.0	If multiple Payment Systems are involved in the Test Session, the TSE Test Session files (with extension .tse) should be grouped in one zip file for the user's convenience.	O
12.0	At any time during the processes above the user shall be able to save his current context by creating or modifying (see below) the (preliminary) TSE Test Session files:	M
12.1	<ul style="list-style-type: none"> <li>If the MODE_Regression indicator is set, the Mode in the TestRunInfo.xml shall be put to MODE_Regression.</li> </ul>	M
12.2	<ul style="list-style-type: none"> <li>Otherwise, if not all questions have been answered, the Mode in the TestRunInfo.xml shall be put to MODE_Question.</li> </ul>	M
12.3	<ul style="list-style-type: none"> <li>Otherwise the Mode in the TestRunInfo.xml shall be put to MODE_Info</li> </ul>	M
12.4	<ul style="list-style-type: none"> <li>The ChangeFlagDate in the same file shall reflect the current timestamp.</li> </ul>	O
13.0	L3TSE shall allow the user to <b>start a TSE Test Session review</b> in the following way:	M
13.1	<ul style="list-style-type: none"> <li>L3TSE allows a user to import a populated TSE Test Session file.</li> </ul>	O
13.2	<ul style="list-style-type: none"> <li>When the TSEC corresponding to the Rule Set Build from the TestRunInfo.xml has not been imported in L3TSE, either a warning shall be issued or the XML's RuleSet.xml shall be used to reconstruct the whole</li> </ul>	M

13.3	<ul style="list-style-type: none"> <li>If the Mode tag in TestRunInfo.xml is MODE_Question then L3TSE shall show the user the first question that needs to get answered and proceed as indicated by the <i>Question Review Process</i> as defined in Requirement 8.             <ul style="list-style-type: none"> <li>If the Mode tag in TestRunInfo.xml is MODE_Regression then L3TSE shall set the MODE_Regression indicator.</li> <li>L3TSE shall <i>present the Information Report</i> for the Terminal Integration Context as defined in Requirement 7</li> </ul> </li> </ul>	M
14.0	L3TSE shall allow the user to update some of the answers from an already created TSE Test Session file and rework the file accordingly. The following procedure is expected to be in place in case the TestRunInfo's RuleSetBuild is still the latest version. (If this is not the case, see the next requirement)	M
14.1	<ul style="list-style-type: none"> <li><i>Start a TSE Test Session review</i> as described in Requirement 13.</li> </ul>	M
14.2	<ul style="list-style-type: none"> <li>If the users want to export the updated Test Session files and the TestRun.xml contains any executed test cases, L3TSE shall raise the question whether the user want to keep previous test results.</li> </ul>	M
14.3	<ul style="list-style-type: none"> <li>The user shall be shown a warning that he is only to allow to keep previous test results if he is sure the outcome of the tests are not depending upon anything that was changed.<sup>3</sup></li> </ul>	M
14.4	<ul style="list-style-type: none"> <li>L3TSE updates the TSE Test Session files             <ul style="list-style-type: none"> <li>Update ChangeFlagDate in TestRunInfo.xml</li> <li>If the user decides to overrule previous test results, then completely refresh TestRun.xml as described in 4.2.3</li> <li>If the user decides to keep previous test results, then update TestRun.xml                 <ol style="list-style-type: none"> <li>Remove all Test Cases that are "Not executed" and no longer applicable</li> <li>Add Test Cases that are applicable and not yet in the TestRun.xml</li> </ol> </li> </ul> </li> </ul>	M

<sup>3</sup> So this is left to the user's responsibility. It might be good to have the following scenarios in mind. Changing the Acquirer's Address Street Name is of course not supposed to invalidate the test results. Enabling Cashback without changing the payment application neither. But changing the payment application itself should require a complete new test run. Changing a limit *may* change expected test outcome (e.g. a previously used amount might no longer be above that limit), so might need a rerun of the test case. Changing the LoA entry might be because of a simple typo, or because a different kernel got used. In the first case, no tests need to be rerun, in the latter case all of them. Having all these scenarios in mind, it might be clear that it cannot be the L3TSE's responsibility to make the judgement, as this is difficult to automate.

	<p>iii. If the MODE_Regression indicator is set, then put status to "Not executed" for all Test Cases that have isRegression set to Yes in RuleSet.xml. Optionally the tool can warn the user that this is going to happen.</p> <ul style="list-style-type: none"> <li>- Completely refresh Selected.xml as described in 4.2.4</li> <li>- Recalculate Manifest file</li> </ul>	
15.0	L3TSE shall allow the user to upgrade their TSE Test Session file to a higher build level. The following procedure is expected to be in place in case the TestRunInfo's RuleSetBuild is no longer the latest version implemented by the L3TSE tool. (If this is not the case, see the previous requirement)	M
15.1	<ul style="list-style-type: none"> <li>• L3TSE shall allow a user to upload a populated TSE Test Session file.</li> </ul>	M
15.2	<ul style="list-style-type: none"> <li>• L3TSE shall ask the user whether or not he wants to upgrade to the latest Build. If not, then the procedure as described in <b>Error! Reference source not found.</b> 14 applies, if yes, then continue as described below.</li> </ul>	M
15.3	<ul style="list-style-type: none"> <li>• The user shall be presented with all now applicable questions not yet answered, or whose answers are no longer valid, like described in the <i>Question Review Process</i>.</li> </ul>	M
15.4	<ul style="list-style-type: none"> <li>• L3TSE then allows the user to export a new TSE Test Session file as described in section 4.2.<sup>4</sup></li> </ul>	M
16.0	L3TSE shall allow the user to copy a TSE Test Session file from one Terminal Integration Context to a slightly different one. The following procedure is expected to be in place:	M
16.1	<ul style="list-style-type: none"> <li>- Start a TSE Test Session review as described in Requirement 13.</li> </ul>	M
16.2	<ul style="list-style-type: none"> <li>- L3TSE then allows the user to export a new TSE Test Session file as described in section 4.2, with one exception: <ul style="list-style-type: none"> <li>- In the TestRunInfo.xml the file shall get a new TrackingNo, and the original TrackingNo should be put in the CopyOf field.</li> </ul> </li> </ul>	M
17.0	The user shall be able to view the contents of the selected test cases <sup>5</sup> :	M

<sup>4</sup> Here it is assumed that previously executed test runs are not reusable. The addition of a single pass criterion to a test case might already make the upgrade of the test results in the new build level fail. It is supposed that the L3TT allows the user to reload the card terminal logs and host logs of a previous run to be validated against the new pass criteria. By doing so the user does not need to rerun the test case itself, only upload previous test logs.

<sup>5</sup> Formally this would be a requirement for the L3TT tool, but it might be considered valuable to have L3TSE to allow a preview to the user.

17.1	<ul style="list-style-type: none"> <li>The selected test cases, on which additionally L3TSE may allow to apply filters indicated by the Attributes</li> <li>The related cards, including the tags defined in the Card File</li> <li>The Test Progress as indicated in the TestRun.xml.</li> </ul>	M
17.2	<ul style="list-style-type: none"> <li>Manual modification of the Test Progress by updating TestRun.xml (for customer that don't want to use an L3TT): <ul style="list-style-type: none"> <li>- Mark pass criteria pass/fail</li> <li>- Mark test cases pass/fail (with automation to do so based upon the set of pass criteria)</li> <li>- Add logs, attachments, observations</li> </ul> </li> </ul>	M
18.0	L3TSE shall be able to import and process XML validation reports (.tser). The following procedure is expected to be in place:	M
18.1	<ul style="list-style-type: none"> <li>L3TSE shall allow a user to upload a validation report (.tser)</li> </ul>	M
18.2	<ul style="list-style-type: none"> <li>L3TSE shall attempt to open the original .tse file with the matching tracking number or guide the user if no matching .tse file is found</li> </ul>	M
18.3	<ul style="list-style-type: none"> <li>L3TSE shall display the information present in the <i>ReviewText</i> (<i>formatted in html</i>)</li> </ul>	M
18.4	<ul style="list-style-type: none"> <li>L3TSE shall update the original .tse file with the info provided in the validation report (.tser)</li> </ul>	M

# 4 EMVCo L3 Detailed Formats

## 4.1 TSE Test Set Files

The machine-readable L3 Test Selection Engine's (TSE) Test Set files are individually provided by the Payment Systems. The files contain the relevant information to allow the L3 TSE to prepare and generate the TSE's Test Session files (one per Payment System), that will be further exported to an L3 Test Tool (L3 TT) engine.

Note that, for streamlining purposes, the individual Test Session files may be consolidated.

### 4.1.1 Test Set File Considerations

#### Bit Definition fields

To automate the TSE data processing (e.g., conditional questions to ask, errors to manage, test cases to select) the context expressions are defined in the **Bit Definition fields** present in some TSE Test Set files. Each context expression is linked to a unique bit of a bitmap representing the Boolean values of the Terminal Integration context. For instance, a Terminal Integration context evaluated via 50 context expressions is associated to a 50-bit long bitmap.

A Terminal integration context having 2 criteria to evaluate the context expression can be mapped as follow:

Context expression	Bitmap position	Context Statement
Interface=Contact	Bit 0	True
Interface=Contactless	Bit 1	True
Term_type=Attended POS	Bit 2	True
Term_type=ATM	Bit 3	False

As the Terminal integration context is an evaluation of the context expressions, it can be represented in this case with a bitmap of length 4, e.g. 0111.

#### Applicability of Context-related Data

The questions, test cases or other data are not always applicable for a given Client's Terminal integration process. They are context-related data. A method of indicating which of them are applicable based on the Terminal Integration context is required.

This is achieved by having two bitmap mask entries for each context-related data.

These masks are identical in length to the Terminal Integration context bitmap described above and each bit represents the corresponding bit in the Terminal Integration context:

- Mask1 - This indicates which bits in the Terminal Integration context should be checked. A "1" indicates that the bit is relevant and a "0" means that the bit can be ignored
- Mask2 - This indicates the expected value of each of the relevant bits in the Terminal Integration context. If all of the bits are as expected, then the record is considered to be in scope

Terminal Integration Context	Mask1	Mask2*	A=Terminal Integration Context and Mask1	In Scope if A = Mask2*
<b>0111</b>	0000	0000	0000	True
<b>0111</b>	0010	0010	0010	True
<b>0111</b>	0010	0000	0010	False
<b>0111</b>	1011	0011	0011	True
<b>011x**</b>	1011	0011	001x	For error: False unevaluated condition Otherwise: True 001 = 101 and 001

\* It is assumed that Mask2 does not set a bit that is not set in Mask1. If this is not the case then Mask2 should be replaced by (Mask1 and Mask2) before processing the applicability.

\*\* Bit 0 still not evaluated. Actually this situation is not supposed to occur when the test plan is set up properly.

The file may contain several occurrences of the same context-related data but with a different set of masks. The file is elaborated so that only one context-related data should be in scope at a time. However it is recommended that test tool vendors exclude any duplicate data.

For readability, both Mask1 and Mask2 are also present in hexadecimal format: HexMask1 and HexMask2.

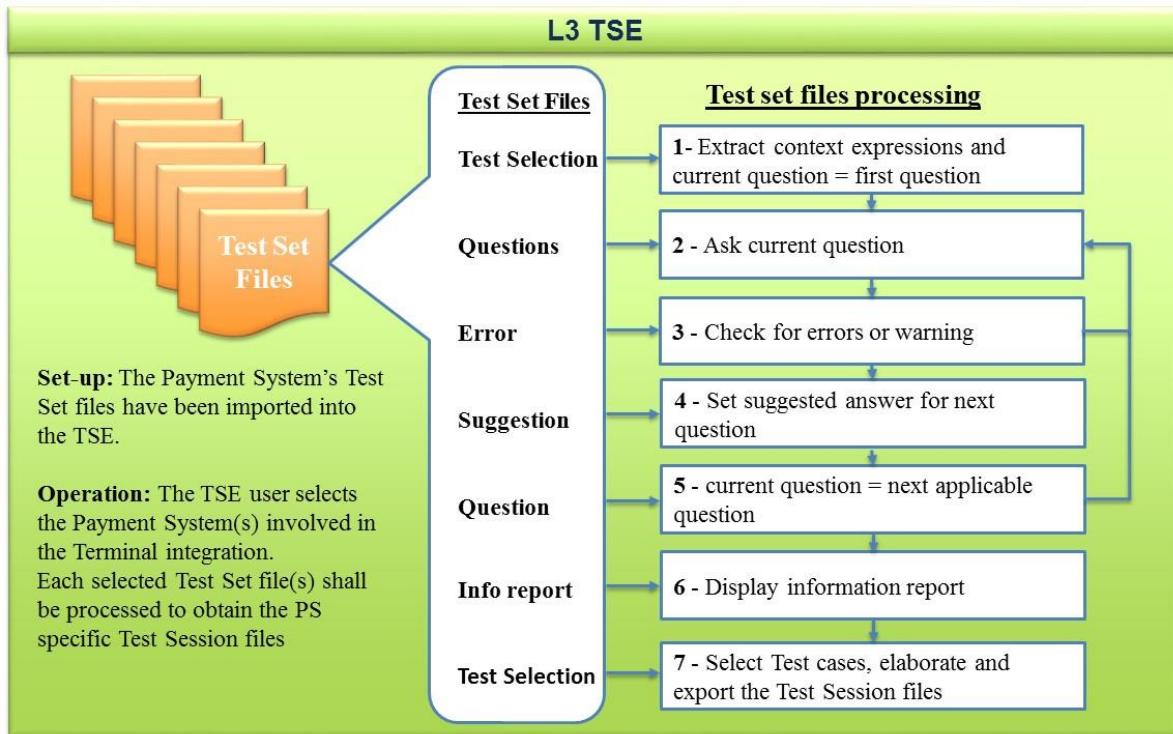
### Processing the TSE Test Set Files

The TSE Test Set files are listed in the table below. They are archived in a .tsec file (a renamed .zip file).

File Type	Format	Context-related data	Purpose
<b>Test Selection File</b>	csv	Yes	Rules for selecting test cases that are in scope based on the Terminal Integration context
<b>Questions Definition file</b>	csv	Yes	A set of questions that needs to be asked in order to determine the Terminal Integration context
<b>Error Definition file</b>	csv	Yes	Identifies invalid responses to questions and provides appropriate error/warning messages
<b>Suggestions file</b>	csv	Yes	Identifies automatic responses to questions based on the responses to previous questions
<b>Information Report file</b>	csv	Yes	Defines information that should be presented to the tester after the questions have been asked
<b>Card file</b>	csv	No	Definition of cards that are to be used in test cases
<b>Test Case file</b>	csv	No	Definition of all test cases
<b>Pass Criteria file</b>	csv	Yes	Definition of pass criteria requirements for all test cases

<b>Test Reference file</b>	html	No	A readable reference document showing a list of all Test Cases and Test Cards
<b>Manifest file</b>	Text	No	Provides a protected list of all the files output and binds them together into a single data set. This file has extension .trcm.

Most of the TSE Test Set files are processed by the TSE in order to generate the L3 Test Session files that will be further exported for L3 testing purposes. This processing is shown in the figure below.



1. The TSE extracts all the context expressions from the Bit Definition fields in the Test Selection file, initialize the bitmap corresponding to the Terminal Integration context by determining the bitmap length (equal to the number of context expressions the Bit Definition fields) and pre-setting the bits to 0. Finally set the current question to the first question in the Questions Definition file.
2. The TSE asks the current question. The answer to the question will typically allow several of the context expressions to be evaluated. For example, if the user selects the response "Attended POS" to a "Terminal type" question then the context expression bit "Term\_type = Attended POS" equates to 1 and the context expression bit "Term\_type = Attended ATM" equates to 0. A partial Terminal Integration context is then available for the next processing step.

3. The partial Terminal Integration context should be evaluated against errors and warnings in the Error definition file. In general, undefined Terminal Integration context bits should be ignored, however in the case of error processing, if a bit is undefined then any error using this bit in its definition is deemed not to match, in other words errors are not reported until the user has had a chance to supply a good answer. If there is an error match, TSE should not allow the user to proceed and they must go back and select a different answer to the current question to generate a partial Terminal Integration context which does not match an error condition. If there is a match with a warning, then the user must be given the option to either proceed with the next question or change their response to the current question.
4. After each question is answered, TSE should examine entries in the Suggestion file. Matching entries may either force a response to a future question, suggest a response for a future question or remove options for a future question. For example, if a user selects an ATM terminal then Online PIN may be forced to be true. Questions that have a forced response should be skipped by the user interface since the user is not allowed to change the value.
5. The partial Terminal Integration context is used to select the next matching question in the Question Definition file to be asked. For example, if the user indicates that they are not using a Service Provider then there is no point asking for the Server Provider's address and contact details. The presence of conditional questions means that even when all in scope questions have been asked there may be bits in the Terminal Integration context that are yet undefined. Moreover, the TSE shall not ask a question that is defined as common between different Payment Systems and for which an answer has already been provided.
6. After all questions have been completed, information records should be selected from the Information Report file and presented to the user. Information records are typically used to provide context specific advice to testers who will perform the test run.
7. The Terminal Integration context is used to select the test cases that are applicable for the test run.

## **TSE Test Set Files Format**

All files are UTF-8 encoded.

The TSE Test Set files that are comma separated files (csv), use the comma (,) to separate the fields, and double quotes to group the content of a field. They contain a header row showing the data field name. The headers can appear in any order.

### **Character Substitutions**

The characters used to structure fields may need to be part of the data itself. In order to allow these character codes a number of substitutions take place when the output file is created. These substitutions should be reversed by tools processing the TSE Test Set files.

**Table 4.1: Character substitution**

<b>Text in Test Set CSV file</b>	<b>Text on user interface</b>
" &&"	"&"
" &ob"	"["
" &cb"	
" &sc"	
" &cl"	".:"
" &cm"	
" &eq"	"="

**Note:** The substitution contains a leading space character so the string "[this&&that=;]" would be converted to the following string after substitutions: " &obthis && &&that &eq &sc &cb".

### **HTML codes in Text field**

TSE file supports the use of HTML codes in text fields that may be used to control the format of the output. For example:

"<b>Purpose:</b><br>The purpose of the test is to check the <b>ARQC</b>"

HTML codes are always specified within angle brackets and with the exception of the <br> tag their support is optional (but highly recommended). L3 TSE tools that do not support HTML must remove all angle bracket codes prior to displaying strings to the user i.e. the <b> tag in the above example must either be interpreted as turning on bold text or completely removed from the string. The text "<b>" must not appear in the output.

Support for the <br> tag (line break) is mandatory and it must cause a new line in the output. Therefore, the above text may be rendered in one of two ways:

Option 1: HTML supported:

**Purpose:**

The purpose of the test is to check the **ARQC**

Option 2: HTML not supported:

**Purpose:**

The purpose of the test is to check the ARQC

**Note:** Some HTML codes include character substitution as describe above, the substitutions should be reversed before HTML tags are interpreted.

Text may include HTML references using the standard HTML link syntax of:  
[topic1](#topic1).

These references indicate placeholders in the help text file the intentions is that tools will implement functionality so that when the topic is clicked the help file is opened at the indicated topic.

An example of a Test Set file .tsec package can be obtained through EMVCo.

### Naming conventions

File Type	Naming convention
<b>Test Selection File</b>	P_N_S_V_selection.csv
<b>Questions definition file</b>	P_N_S_V_questions.csv
<b>Error definition file</b>	P_N_S_V_error.csv
<b>Suggestions file</b>	P_N_S_V_suggest.csv
<b>Information Report file</b>	P_N_S_V_info.csv
<b>Card file</b>	P_N_S_V_card.csv
<b>Test Case file</b>	P_N_S_V_cases.csv
<b>Pass Criteria file</b>	P_N_S_V_pass_criteria.csv
<b>Test Reference file</b>	P_N_S_V_test_reference.html
<b>Manifest file</b>	P_N_S_V_manifest.tricm

TSE Test Set files are versioned using

- P: EMVCo L3 Payment System's Identifier,

Payment System Identifier	Payment System
01	American Express
02	Discover
03	JCB
04	Mastercard
05	UnionPay
06	Visa
07 ... 99	Reserved for future Payment Systems

- N: Payment System's Test Set file name, a Payment System proprietary information that may describe the use of the Test Set files.
- S: Series number, a unique and Payment System proprietary identifier that represents a particular data set for a dedicated purpose (e.g. Contact only testing).
- V: Build number. This represents the version of the file within the series. Higher version numbers within the same series identify newer versions of the same data set. The L3 TSE tool shall always support the latest version of data files.

Example of a TSE Test Set files version number: *04\_contact\_01\_10* identify the TSE Test Set files series n°01 and Build version 1.0 from Mastercard.

#### 4.1.2 Test Selection File Format

**Purpose:** Rules for selecting test cases that are in scope based on the context of the Terminal integration. Undefined bits in the Terminal Integration context (caused by conditional question not be answered) should be considered as being zero for the purposes of test case selection.

Each record provides the selection rule of a given test case. Test cases detailed description can be found in the Test Case file.

Data field	Description
<b>Mask1</b>	See Applicability of Context-related Data
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>TestCase</b>	Unique identifier of the test case. Provided by the Payment System. Each identified Test Case is described in the Test Case Definition file.
<b>Context Expression 1 .. n</b>	One entry for each context expression that must be evaluated to define the Terminal integration context.  There is a dedicated field for each bit in the bitmap that provide the definition of a given context expression. The number of entries shall be equal to the number of bits in the mask1 and mask2. To allow the HexMask to be created, the number of context expressions should be a multiple of 8. For that reason some data fields may be added with the name " <b>unused</b> "

### 4.1.3 Question Definition File Format

**Purpose:** A set of questions that needs to be asked in order to determine the Terminal Integration context. Each question has a unique identifier that corresponds to the criteria name in the Bit Definition fields of the test selection file. There are two types of questions:

- **Mandatory:** Questions that will be asked irrespective of the Terminal Integration context.
- **Conditional:** Question that will be asked in accordance with the Terminal Integration context

The first question is always mandatory.

#### **Common Questions:**

Some of the questions and related answers might be common to several Payment Systems (e.g. some administrative information, some terminal configuration features, etc.) The TSE shall be able to identify the questions that have already been responded to, retain the related answers, thus avoiding redundant questions being asked again and applying known answers to each Payment System's test case selection or other data processing. To achieve this requirement, common questions will be identified as those

- i.sharing a same criteria name,
- ii.sharing the same question definition, and
- iii.featuring an attribute called 'Common' being in the Question Definition file.

This is the only context in which the TSE uses information pertaining to multiple test sessions. In all other cases, the test sessions corresponding to different Payment Systems are managed separately.

Each record provides the definition of a given question

Data field	Description
<b>Mask1</b>	See Applicability of Context-related Data
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>GroupNo</b>	Left blank in case of a stand-alone question N = This question belongs to the group N and should be asked with the other group N's questions to the user on a single screen
<b>GroupPrompt</b>	Free text used to introduce the Group edit screen to the user
<b>GroupHelp</b>	Help text made available to the user in HTML format
<b>GroupMode</b>	Reserved for future use
<b>Name</b>	Unique name of the question being one of the criteria names used in the context expressions
<b>Type</b>	Define the type of the question which is related to: <ul style="list-style-type: none"><li>• varBoolean - Boolean value. True or False</li><li>• varNumber - Numeric value.</li><li>• varList - mutually exclusive list of strings</li><li>• varSet - list where more than one option can be selected.</li><li>• varString - free text string.</li></ul>

<b>Allowed</b>	For list and set items this field provides a list of allowed values in the format: [value1,value2, ... ,valueN] Example: [ATM,Bank Branch Terminal,Attended POS, Unattended POS,On-board Terminal,Mobile POS (MPOS)]
<b>Attributes</b>	Holds a list of attributes, separated by a semi-colon: <ul style="list-style-type: none"> <li>Common - defines if the expected answer is common to all Payment System to avoid redundant question processing.</li> <li>Regression - indicates that changing this question should issue the test cases that are highlighted as regression test cases to be retested</li> </ul>
<b>Prompt</b>	Text to be used to ask the question in HTML format
<b>Help</b>	Help text made available to the user in HTML format
<b>Mode</b>	Layout indication for the question presentation to the user. Multiple modes may be present, separated by a semi-colon <ul style="list-style-type: none"> <li><b>optional</b> - indicates that the field is optional. If this is not present, then the field should be assumed to be mandatory</li> <li><b>input_mode=drop_down</b> - indicates a preference for the user interface to use a dropdown list for options rather than a long list</li> <li><b>input_mode=drop_down:N</b> - as above but indicates that the dropdown list should have space for N items.</li> <li><b>input_mode=multiline:N</b> - indicates that the user interface should allow up to N lines of text to be entered (where N is in the range 1 to 9)</li> <li><b>input_mode=multi_select</b> - applicable for varSet questions only and indicates that two lists should be presented, a list of unselected items and a list of selected items with a mechanism to move items between the two lists</li> <li><b>input_mode=multi_select:N</b> - as above but indicates that the lists should have space for at least N items to be displayed.</li> <li><b>validation=email</b> – validates that the entry is an email address, i.e. a string satisfying regex <code>.*@.*</code></li> <li>•</li> </ul>
<b>Groups</b>	Define group values if group names are used in the Allowed field [group_name_1=value1,value2,valueN]; [group_name_2=value1,value2,valueN] These groups are not shown to the user, but used to evaluate context expressions that use the <i>in</i> operator.
<b>Context Expression 1 .. n</b>	Please refer to the Test Selection file description. The allocation of bits to criteria in the file is identical to the allocation in the Test Selection file.

#### **4.1.4 Error Definition File Format**

**Purpose:** Identifies invalid responses to questions and provides appropriate error/warning messages. The error definition file uses the mask mechanism to identify error or warning conditions that are in scope.

Each record provides the definition of a given error

Data field	Description
<b>Mask1</b>	
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>Name</b>	Internal ID for the error
<b>ErrorType</b>	Identifies the issue level: <ul style="list-style-type: none"><li>• errError - Indicates that this is an error condition that should be resolved before selecting test cases.</li><li>• errWarning - indicates that this is a warning condition. The user may change a response to a question to resolve it, but should be allowed to continue to test case selection.</li></ul>
<b>Message</b>	Text error message to be displayed in HTML format
<b>GotoQuestion</b>	Name of the question that must be changed in order to resolve the error condition.
<b>Context</b>	Please refer to the Test Selection file description.
<b>Expression</b>	The allocation of bits to criteria in the file is identical to the allocation in the Test Selection file.
<b>1 .. n</b>	

#### 4.1.5 Suggestion Definition File Format

**Purpose:** Provide rules for changing the answer to the current question based on previous question responses. Three options are available:

- Suggest answers for the question - which the user can change
- Force a particular answer for the question - which the user can't change
- Remove options from list/set questions that are not relevant

The removal option is used in situations where a choice in a list (or set) is no longer appropriate. For example, in certain regions it may not be appropriate to include "SDA" in the list of possible Cardholder Authentication Methods.

It may be the case that several suggestion rules are in scope at the same time for a given question. In which case the following rules apply:

- iv. Rules are implemented in the order listed so that the last rule applies - e.g. for list questions with several suggested values the last suggested value should be used
- v. Forced values will always override suggested values regardless of the order of rules in the file
- vi. Suggestions and Forced values for set questions are accumulative e.g. if there are two active suggestions for a set question then both values should be proposed
- vii. Removals for list and set questions are accumulative

Each record provides the definition of a given suggestion

Data field	Description
<b>Mask1</b>	See Applicability of Context-related Data
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>QuestionName</b>	Unique name of a question in the Question definition file
<b>SuggestionType</b>	The type of suggestion may have the following values: <ul style="list-style-type: none"><li>• force</li><li>• suggest - Contains a suggested list of values for the question</li><li>• remove - Contains a list of values to be removed from the Allowed values list in the Question definition file</li></ul>
<b>Values</b>	List of values to suggest, force or remove in the following format: [value1][value2][value3]
<b>Context Expression</b>	Please refer to the Test Selection file description. The allocation of bits to criteria in the file is identical to the allocation in the Test Selection file.
<b>1 .. n</b>	

#### 4.1.6 Information Report File Format

**Purpose:** Define information that should be presented to the tester after the questions have been asked. This information will give recommendations and instructions for a given Terminal Integration context. For example, depending on the answers provided, the user may need to be informed about particular Terminal Action Codes to be used.

The intention is that TSE should process the information report file after all applicable questions have been answered and assemble the information into a separate report that is presented to the user.

Data field	Description
<b>Mask1</b>	See Applicability of Context-related Data
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>HeadingLevel</b>	Level from 1 to 5
<b>Name</b>	Internal ID for the Information entry
<b>SectionHeader</b>	Heading text of the heading level in HTML format
<b>Information</b>	Text information in HTML format. To be displayed in the body of the report underneath the section header. The information text may also contain placeholders in the text string of the form \$question_name\$. The placeholders should be replaced with the value given for the named question.
<b>Context Expression</b> <b>1 .. n</b>	Please refer to the Test Selection file description. The allocation of bits to criteria in the file is identical to the allocation in the Test Selection file.

#### 4.1.7 Card File Format

**Purpose:** Definition of the cards that are to be used in test cases

Each record provides the definition of a given card

Data field	Description
<b>ID</b>	Unique card Identifier. Provided by the Payment System.
<b>Name</b>	Friendly name of the card
<b>Description</b>	Text description of the card technical details
<b>History</b>	<ul style="list-style-type: none"><li>• Major - Integer - major version number incremented with each new major version</li><li>• Minor - Integer - minor version incremented with each new minor version</li><li>• Description - this is a textual description of the change made</li></ul> <p>The version numbers are particularly significant to identify the test(s) that should be performed again in case of major change(s).</p>
<b>Tags</b>	<p>Detail a "tag" together with its associated value:</p> <ul style="list-style-type: none"><li>• Tag - Name of the tag e.g. "PIN", "PAN", "Brand" etc.</li><li>• Value - Value of the tag e.g. "4321", "5432-0000-0000-0000", "PSx Brand"</li></ul> <p>The tag name can be hierarchical and use a "dot" notation to separate levels in the hierarchy. For example, a card may have the tags "Application1.PIN" and "Application2.PIN".</p> <p>This is represented in the following way:</p> <pre>[Tag=PAN:Value=nnnnnnnnnnnnnn];[Tag=Brand:Value=PSx];[Tag=PIN:Value=4315]</pre>

#### 4.1.8 Test Case File Format

**Purpose:** Provide details about all test cases listed in the Test Selection file

Each record provides the definition of a given test case

Data field	Description
<b>Name</b>	Unique name that corresponds to a test case name in the Test Selection file
<b>History</b>	<ul style="list-style-type: none"> <li>• Major - Integer - major version number incremented with each new major version</li> <li>• Minor - Integer - minor version incremented with each new minor version</li> <li>• Description - this is a textual description of the change made</li> </ul>
<b>Cards</b>	Unique card identifier that corresponds to the card identifier in the Card file. A Tester may have the choice to select one card in a list: [ Card=<test card1>];[ Card=<test card2>]
<b>Objective</b>	Text description of the test objective in HTML format
<b>Configuration</b>	Text description of the configuration that is required before test execution
<b>Applicable</b>	Text description indicating when the test case is applicable in HTML format
<b>Notes</b>	Text description of notes relates to the test
<b>Context</b>	For reference only
<b>Actions</b>	Text description of the action(s) that the tester shall take Format: [Action=<action1>];[ Action=<action2>];[ Action=<actionn>]
<b>Requirements</b>	<p>Details the documents or sections of document that is relevant for the test being performed:</p> <ul style="list-style-type: none"> <li>• Document – Identify the document</li> <li>• Section – Identify the section in the document</li> </ul> <p>Format: [Document=&lt;document1&gt;:Section=&lt;section1&gt;];[Document=&lt;document2&gt;:Section=&lt;section2&gt;]</p>
<b>Attributes</b>	<p>Hold a list of attributes for possible filtering purpose</p> <p>Format: [attribute_1];[attribute_2];[attribute_n]</p> <p>For example, online tests could be tagged with the attribute "online" and offline tests with the attribute "offline" - this would allow users to select only the offline or online tests by filtering on these attributes.</p>
<b>isRegression</b>	<p>Indicate if the test is for regression testing purpose:</p> <p>Yes – The test should be repeated when the user changes the test selection question(s) after partial or full test completion</p>

#### 4.1.9 Pass Criteria File Format

**Purpose:** Provide the pass criteria checks that should be performed on each test after the actions have been performed. The file is structured so that tools can automatically perform most of the check by examining card and network logs that are generated during the testing process. Alternatively, the file also allows a user interface to guide a tester through the required pass criteria steps.

Some of the pass criteria checks may be conditional and only applicable in certain circumstances, for example, the check for a printed receipt should only be performed if the terminal is equipped with a receipt printer.

The file incorporates the mask mechanism to indicate when a particular pass criteria check should be performed for a test.

It is anticipated that tools will first extract the relevant tests using the test selection file and then extract the in-scope pass criteria checks for each test by using the data in this file.

Some tests may consist of multiple steps, each step will have an incrementing step number and will be recorded on a separate line in the CSV File.

**Multiple records:** Each record provides the definition of a given Pass criteria

Data field	Description
<b>Mask1</b>	See Applicability of Context-related Data
<b>Mask2</b>	
<b>HexMask1</b>	
<b>HexMask2</b>	
<b>Test</b>	Unique test case identifier the check is linked to
<b>Commentary</b>	Free text describing the purpose of the check
<b>CheckNumber</b>	The definition is provided in <b>chapter 4.3</b>
<b>StepOfCheck</b>	
<b>IsMandatory</b>	
<b>DataItem</b>	
<b>DataFormat</b>	
<b>DataItemName</b>	
<b>Operator</b>	
<b>Value</b>	
<b>ValueItemName</b>	
<b>ActionIfTrue</b>	
<b>ActionIfFalse</b>	
<b>Context Expression</b>	Please refer to the Test Selection file description.
<b>1 .. n</b>	The allocation of bits to criteria in the file is identical to the allocation in the Test Selection file.

#### **4.1.10 Test Reference File Format**

**Purpose:** The test reference file is an HTML version 4 file providing a complete list of all tests cases and test cards that may be used in the testing process. The intention is that tools can display a textual representation of test cases on tester's request.

It consists at least of three main sections:

Table of Contents - providing hyperlinks to each test case and test card

Test Definitions - description of each test case with hyperlinks to cards that are used

Card Definitions - description of each card definition with links to test cases where the card is used

#### 4.1.11 Manifest File Format

**Purpose:** The manifest file is a text file with extension .tricm that:

- Allows to check the integrity of the TSE Test Set files in **csv** format
- Binds these files together into a complete data set.

Test tools may use the manifest file to control the integrity of the TSE Test Set files or select the appropriate set of files when several manifest files are present. The TSE shall use the latest version of a given series.

**File format:**

Manifest file is a list of parameters name and corresponding values as shown in the example below: (tabs added for readability)

PSI	=1
Series	=0
Build	=140
Name	=DemoPlan
Mode	=EMVCoL3
Selection_File	=02_DemoPlan_0_140_selection.csv: LEy6saV522S2l8h+jwPxcKYW7I8=:
Scenario_File	=02_DemoPlan_0_140_test_case.csv: MXbW1fj+qZYAUa1fvJbOohVjGxs=:
Question_File	=02_DemoPlan_0_140_question.csv: j13wfAVBV3rr/p+0A3ZVTSqWNC0=:
Error_File	=02_DemoPlan_0_140_error.csv: V9bh7yrF8s0KMgStFJ0kbmenxbo=:
Suggest_File	=02_DemoPlan_0_140_suggestion.csv: nGgXbqGRWySNmyC2x9j+LmRQ8bl=:
Pass_Criteria_File	=02_DemoPlan_0_140_pass_criteria.csv: QRPu96j3zODQvlxSAGITUObh7WY=:
Card_File	=02_DemoPlan_0_140_card.csv: P92BJUcBHRqEinwxN50DvGNwUT4=:
Information_File	=02_DemoPlan_0_140_information.csv: Kmfr/M5S6TCAc2svvvZ4t963NiM=:
Test_Reference	=02_DemoPlan_0_140_test_reference.html:eI4Pj5tA0EHL8XHYOtbbseLe7qg=:
Signature	=vzFajPLsUNK=

## 4.2 TSE Test Session Files

The machine-readable TSE Test Session files are generated by the L3 TSE Tool. They contain the relevant information for the L3 Test Tool and the tester to perform the selected test cases and checks and then provide the test results, the related logs and tester observations.

The TSE Test Session files are the following XML files archived in a **.tse file** (a renamed .zip file):

- TestRunInfo.xml – This file contains management information about the test run as a whole.
- RuleSet.xml – This file contains static information about the rule set being used by TSE for the test run. This is an XML copy of the TSE Test Set csv data used to elaborate the Test Session files.
- TestRun.xml – This file contains dynamic information about the test run, including question responses, test results, log file names and test observations. This file will be dynamically updated by L3 Test Tools to provide test results, the related logs and tester observations as part of the machine-readable Test Report.
- Selected.xml – This file provides a list of in-scope test cases and test steps.
- Manifest.txt – Optional - Provides a protected list of all the files output and binds them together into a single data set

Each XML file has a single top level tag called `<L3tse>` that contains child elements used to hold the data that is being managed by TSE and L3 Test Tool. The top level `<L3tse>` tag contains a *file version* attribute that should be set to "1", together with an *originator* attribute that should be used to indicate which tool created the file.

For data integrity checking purpose all files include a checksum in an XML comment on the last line of the file.

The comment contains the Base64 encoded SHA-1 hash of the XML file excluding the comment line itself. When importing files, the L3 TSE or L3 Test Tool will warn the user if this hash value is not correct. The intention is that automated tools will always maintain the correct checksum but if the file is manually changed without updating the checksum then a warning should be displayed.

A typical header & trailer example is detailed below:

```
<?xml version="1.0" encoding="utf-8"?>
<L3tse file_version="1" originator="L3 TSE NNNNN">
...
</L3tse>
<!--pEkDJS1je98OIPki5w/fDQXWKjE=-->
```

#### 4.2.1 TestRunInfo.xml

Section - Block tag - Data field tag	Description	L3TSE Requirement
<TestRunInfo>	Provide information of the test session to the L3TSE for testing session status management purpose	
...<Context>	Single entry	
....<TrackingNo>	Unique identifier of the Terminal Integration test session	<Mode from Manifest file>_<PSI>_<timestamp in milliseconds>
....<CloneOf>	When present holds the TrackingNo of the test run the clone is from.	
....<CopyOf>	When present holds the TrackingNo of the test run the copy is from.	
....<MachineID>		Optional <sup>6</sup>
....<RuleSetName>	Capture the version used when the test run was originally created. L3TSE can use this information to determine whether the test run is using previous versions of Test Plans	Take Name from Manifest file
....<RuleSetSeries>		Take Series from Manifest file
....<RuleSetBuild>		Take Build from Manifest file
....<RuleSetMode>	Purpose of the Test run (RuleSet) = EMVCo L3 Terminal Integration	Take Mode from Manifest file
....<TestReferenceFile>	Files containing test reference information, help information and CSS style sheets. Stores a link to the filename that contains the information.	Optional
....<HelpFile>		Optional
....<DocStyleFile>		Optional
....<ScreenStyleFile>		Optional
....<ChangeFlagDate>	At package creation, TSE sets this date to the current date and time.	Creation timestamp
....<Mode>	Current status of the test run describes by one of the following values: MODE_Question Questions still need to be answered to determine the tests that are in scope MODE_TestResult All questions have been answered and test results are being recorded MODE_Info All questions have been answered and test information is being displayed MODE_Regression Questions have been changed and the user should be prompted to clear regression test results	MODE_TestResult

<sup>6</sup> Optional fields may be filled with content to the L3TSE's discretion. There is no guarantee that this will work when opened in another vendor's L3TSE.

Section - Block tag - Data field tag	Description	L3TSE Requirement
....<FilterActive>	Use these tags to keep track of Filtering by the user.	Optional
....<FilterOptions>		Optional
....<SelectedTests>	User discretionary sub-selection of test cases.	Optional

#### 4.2.2 RuleSet.xml

The RuleSet file is a XML representation, copying data from the TSE Test Set CSV files described in chapter 4.1. As such it contains static data that remains unchanged during the test execution.

- Each CSV file has a dedicated section in the RuleSet XML file (XML File Tag in the table below).
- Each CSV record has a dedicated sub-section within a section (XML Record Tag in the table below).
- Each CSV data has a dedicated Tag within a sub-section (Data Field Tag) holding the same name.
- The name of XML Data Field tag within each record is identical to the column names in the corresponding CSV file, however some redundant columns are omitted.

CSV File	XML File Tag (Section)	Purpose
<b>Test Selection File</b>	<Bitmap> ..<Bit> ....<Criteria>	List the Criteria names in the right Bitmap order.
<b>Test Selection File</b>	<Selection> ..<Rule> ....<Mask1> ....<Mask2> ....<TestCase>	Rules for selecting test cases that are in scope based on the context of the project. Reflect Mask1, Mask2 and TestCase from Selection file
<b>Question Definition File</b>	<Questions> ..<Question> ....<mask1> ....<mask2> ....<groupno> ....<groupprompt> ....<grouphelp> ....<groupmode> ....<name> ....<type> ....<allowed> ....<groups> ....<prompt> ....<help> ....<mode>	A set of questions that needs to be asked in order to determine the project context. All come from the Question definition file.
<b>Error Definition File</b>	<Errors> ..<Error> ....<mask1> ....<mask2> ....<ErrorType> ....<message> ....<GotoQuestion>	Identifies invalid responses to questions and provides appropriate error/warning messages.

<b>Information Report File</b>	<InformationReport> ..<Message> ....<Mask1> ....<Mask2> ....<HeadingLevel> ....<SectionHeader> ....<Information>	Defines information that should be presented to the tester after the questions have been asked.
<b>Card File</b>	<CardRecords> ..<card> ....<id> ....<name> ....<description> ....<history> ....<tags>	Definition of cards that are to be used in test cases.
<b>Test Case File</b>	<Tests> ..<Test> ....<name> ....<history> ....<cards> ....<objective> ....<configuration> ....<applicable> ....<notes> ....<context> ....<actions> ....<requirements> ....<attributes> ....<isRegression>	Definition of all test cases.
<b>Pass Criteria File</b>	<Checks> ..<Check> ....<mask1> ....<mask2> ....<Test> ....<Commentary> ....<CheckNumber> ....<StepOfCheck> ....<IsMandatory> ....<DataItem> ....<DataFormat> ....<DataItemName> ....<Operator> ....<Value> ....<ValueItemName> ....<ActionIfTrue> ....<ActionIfFalse>	Definition of pass criteria requirements for all test cases.
<b>Suggestion File</b>	<Suggestions> ..<suggestion> ....<mask1> ....<mask2>	Identifies automatic responses to questions based on the responses to previous questions

```

....<QuestionName>
....<SuggestionType>
....<Values>

```

## Substitutions

As described in the Test Set files definition, the CSV data contains various substitutions so that characters can be used to delimit structured fields. See Table 4.1. The substitutions don't need to be passed on to the XML file, e.g. "&ob" can be replaced by "[" in the XML. It might be wise to keep separators like &cm, &sc and &cl as such in the XML, in order to avoid that lists get ambiguous. However, XML has its own set up substitution rules that are applied:

**Table 4.2: XML character substitution**

Text in Test Set CSV file	Text in Test Session XML file
&	&amp;
>	&gt;
<	&lt;
'	&apos; (only when needed)
"	&quot; (only when needed)

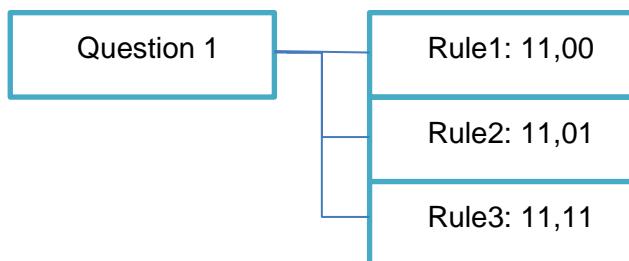
So &cm in the CSV becomes &amp; cm in the XML.

## Processing Records with multiple mask values

The TSE Test Set files often contains multiple records for the same item, for example, if a question can be applicable in different circumstances then the question will appear multiple times in the output and each entry will have a separate set of mask values. For example:

Mask 1	Mask2	Question Details
11	00	Question1 ...
11	01	Question1 ...
11	10	Question1 ...

Rather than modelling this with multiple records within the application logic, tools that need to process the CSV information are recommended to create a single record with multiple mask rules:



The approach is recommended for all records that contain multiple masks since it typically simplifies tool design, for example, answers are linked with a single question record rather than multiple records.

### 4.2.3 TestRun.xml

This XML file is used by the L3 TT to report the test session results and provide the logs file names and directory paths as well as the tester observations and answers to provided questions. The file is read and updated from L3 TT perspective. The table below describes how the initial TestRun.xml can be created by the L3 TSE tool. It should only reflect the applicable test cases and pass criteria. Inapplicable ones should be excluded from the XML.

Section	Create
<Results>	
...<Test>	
....<TestName>	
....<MajorVersion>	
....<CheckNo>	
....<StepNo>	
....<Result>	“not tested”
....<Updated>	XML Creation Timestamp
<Log files>	
...<Entry>	
....<Test>	RuleSet XML: <Tests><Test><name>
....<Files>	
....<Updated>	XML Creation Timestamp
<Attachments>	
...<Entry>	
....<Files>	
<Observations>	Leave empty: <Observations/>
...<Test>	
....<TestName>	
....<MajorVersion>	
....<Observation>	
....<ReviewComments>	
>	
....<Updated>	
<Answers>	
...<Response>	
....<Name>	<Questions><Question><name>
....<Answer>	The answer as given during the L3 TSE process to this question
....<Updated>	Either timestamp when the question was answered, or timestamp when the TestRun.xml file was created. Both are acceptable.

During execution, the L3 TT tool will update the TestRun.xml file in the following way

Section	Update
<Results>	

Section	Update
..<Test>	<i>Provide the check results of the test session</i> <b>Multiple entries.</b> <i>Give the result of a given Pass Criteria</i>
....<TestName>	
....<MajorVersion>	
....<CheckNo>	
....<StepNo>	
....<Result>	<b>“Fail” or “Pass”</b>
....<Updated>	Update Timestamp
<Log files>	<i>List the log files that have been recorded during the test session.</i> <i>The log files can be a Card Terminal log or an Online message as defined in this document.</i> <b>Multiple entries.</b> Specify the logs names and directory path which are linked to a given Test. Should be present for each Test Case in the Test Case File.
..<Entry>	
....<Test>	
....<Files>	Comma separated of quoted Log file name(s) + directory path e.g. “c:\titf\Logfile1.txt”, “c:\titf\Logfile2.pdf”
....<Updated>	Update Timestamp
<Attachments>	<i>List the reference documents that could be helpful for the test session</i>
..<Entry>	
....<Files>	Comma separated of quoted file name(s) + directory path e.g. “c:\titf\file3.txt”, “c:\titf\file4.pdf”
<Observations>	<i>Provide the tester observations for the test session. Give the test observation for a given Test Case</i>
..<Test>	
....<TestName>	RuleSet XML: <Tests><Test><name>
....<MajorVersion>	RuleSet XML: <Tests><Test><history> Major tag value
....<Observation>	Observation as entered by the Test Analyst in the L3 TT tool
....<ReviewComments>	
....<Updated>	Update Timestamp
<Answers>	<i>List the answers to the questions in &lt;Questions&gt; section in the RuleSet XML file. Answer(s) supplied for a given question</i>
..<Response>	
....<Name>	
....<Answer>	
....<Updated>	

#### 4.2.4 Selected.xml

This XML file will be used by the L3 TT to obtain the list of the Test Cases to perform as well as the identification of the Pass Criteria that apply to each Test Case. The file is read-only from L3 TT perspective.

Section	Description
<InScope>	<i>List of the test cases that TSE has defined in Test Session scope</i>
..<Test>	

```
....<name>                               RuleSet XML: <Tests><Test><name>
....<status>                                "In-Progress"
<ActivePassCriteria>
..<Step>
....<TestName>
....<MajorVersion>
....<CheckNo>
....<StepNo>
```

#### 4.2.5      Manifest file

This is a plain copy of the Manifest file from the TSE Test Set. The file extension is .txt.

## 4.3 Tool Pass/Fail Automation Criteria

### Test Case Pass/Fail Criteria Principles

Below is the list of principles for defining the pass/fail criteria for test cases associated with the L3 testing processes. These principles have been defined in an effort to provide clear and consistent definitions of pass and fail criteria, which ultimately helps to eliminate any uncertainties among Client Testers.

**Note:** These principles also align with those defined in the *EMVCo Type Approval Terminal Level 2 Test Cases Version 4.3c – November 30<sup>th</sup>, 2013* document.

- **Language:** Universal language (British English) should be used.
- **Definitions:** Should use pre-defined Abbreviations and Notations, and defined in the Common Terminology Glossary (Annex A).
- **Consistency:** Should be consistent with the other parts of the same L3 test case ("Objective", "Test Procedures" etc.) etc.) as well as the L3 Pass criteria.
- **Pass criteria:** One or multiple Pass criteria for one test case are allowed.
- **Expressions:** "If", "and" or "either" conditions should be avoided where possible, and also avoid vague expressions such as "may", "should", etc. (use of "shall" is preferred).
- **Independence:** Pass criteria shall be comprehensible independently without having to be read in conjunction with the other related documents.
- **Clarity:** Ensure instructions are clearly defined, e.g. perform a transaction for "10" and, if instructed, enter a PIN value of "XXXX".

Since most of the chip tool vendors have historically used their own internal representation syntax to determine the pass criteria for each Payment System's test cases, the TITF has agreed to utilize a common syntax to formally specify the checks that are to be performed on each test case to determine the outcome. This approach will enable the tools to automatically execute the validation checks required in both the Card/Terminal Log and the Acquirer Host Message and as defined in the Test Set Template (refer to the chapter 4.1).

The key principles of the common syntax are as follows:

- **Machine-readable:** The common syntax will be provided to the tool vendors in machine-readable form for implementation. The file that embeds the common syntax will be referred to as the pass criteria file.
- **Format:** The pass criteria are defined in the section 4.1.9 Pass Criteria File Format.
- **Test Session Alignment:** The fields included in the syntax should align with corresponding fields in the Test Session Template.

**Note:** no specific method, process or tool for creating the syntax is being prescribed here. The creation method will solely be at the discretion of each Payment System.

Below are the agreed upon fields and definitions for the EMVCo L3 syntax.

**Table 4.3: List of Syntax Fields and Definitions**

Tag in TSE file	Description
Test	Provides the identifier of the test (as defined in the TSE file) to which the pass criteria checks are linked. Tools may extract all the relevant checks for a particular test by extracting all entries from the pass criteria file which have a matching name for the given test.
Commentary	Free text commentary field describing, at a high-level, the purpose of the check being performed.
CheckNumber	Each separate pass criteria check is allocated a unique incrementing check number. If a test has multiple steps then each step of the check has the same check number.
StepOfCheck	Each step of an individual check is allocated a unique incrementing step number. The combination of test, check number and step number can be used to uniquely identify a particular pass criteria check action.
IsMandatory	Field may have one of two possible values: MANDATORY - The check must be performed SUGGESTED - The check is suggested but may be skipped if, for example, manual verification is taking place The intention is that tools which automatically perform pass criteria checks should complete both the mandatory and suggested checks. In the case where the checks are being performed by an operator manually checking logs then the work can be made less onerous by skipping the suggested tests.
DataItem	This field gives the full name of data item to be checked. The field either contains the fixed text "MANUAL" to indicate a manual check is present in the value field or the name of a data element that needs to be extracted to perform this test.  The name is a hierarchical description of the data item using a "dot" notation to separate levels. The intention is that the description uniquely and unambiguously describes what data need to be checked using a name that can be understood by automated tools.  At the top level the notation starts with: NET.XXXIN - The data to be checked is from a Network message <ul style="list-style-type: none"><li>- XXX = the network message identifier. The ? character may be used as a wildcard to indicate any valid character. For example, "NET??10" indicates any network message that has an identifier ending in 10.</li></ul> When a wildcard is being used, tools looking for data items should search all candidate network messages for the specified item until the item is found. <ul style="list-style-type: none"><li>- IN = optional indicator that the data item represents instance N of the message</li></ul>

Tag in TSE file	Description
	<p>APDU.CCIIP1P2IN - The data to be checked is from an APDU that has been exchanged with the card</p> <ul style="list-style-type: none"> <li>- CC =Hex CLA byte</li> <li>- II = Hex INS byte</li> <li>- P1 = Optional P1 byte</li> <li>- P2 = Optional P2 byte</li> <li>- IN = optional indicator that the data item represents instance N of the message. The last instance is represented by the keyword "IL".</li> </ul> <p>The APDU definition may also use the wildcard ? character to match any valid hex nibble. Tools processing the syntax must search all matching APDUs when searching for wildcard data elements. This also applies when the P1/P2 parameters have not been defined since there may be multiple matching records that could hold the data element.</p> <p>ATR.MODE - The data to be checked is from the "Answer To Reset" from the card</p> <ul style="list-style-type: none"> <li>- MODE = "COLD" - indicates the ATR from the card cold reset</li> <li>- MODE = "WARM" - indicates the ATR from the card warm reset</li> </ul> <ul style="list-style-type: none"> <li>• MANUAL - This is a manual check that can't be completed automatically. The value column (see below) gives a textual description of the check to be performed</li> </ul> <p>Some example uses of the top level notation would be:</p> <p>NET.0100 = The authorization request in the network transaction log</p> <p>NET.0100I2 = The second authorization request in the network transaction log from the start of this test case</p> <p>APDU.80AE = Generate Application Cryptogram Command</p> <p>APDU.80AEI2 = Second Generate Application Cryptogram Command</p> <p>APDU.00B201 = Read Record 1 Command</p> <p>APDU.00B2 = Any read record command (since P1 is not specified)</p> <p>ATR.WARM = Bytes from the warm Answer to Reset</p> <p>The top level notation is further qualified by additional "dot" elements that further qualify the data item. In the case of NET data items the further qualification is as follows:</p> <ul style="list-style-type: none"> <li>• NET <ul style="list-style-type: none"> <li>- DE.NNN - indicates data element NNN should be checked <ul style="list-style-type: none"> <li>◦ SE.NNN - indicates that sub-element NNN should be checked <ul style="list-style-type: none"> <li>▪ BYTE.OFFSET-LEN</li> <li>▪ TAG.TAG_NO</li> </ul> </li> </ul> </li> <li>- BYTE.OFFSET-LEN</li> <li>- TAG.TAG_NO <ul style="list-style-type: none"> <li>◦ BYTE.OFFSET-LEN</li> <li>◦ TAG.TAG_NO</li> </ul> </li> </ul> </li> </ul>

Tag in TSE file	Description								
	<p>An additional DE qualification is added to indicate which data element in the network message should be checked. Individual bytes of this data element may be checked by using the .BYTE notation, alternatively if the data element is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation.</p> <p>The OFFSET given in the BYTE notation is a byte offset in the data when it is considered as a hexadecimal string. The LEN parameter is only present if the value is not 1.</p> <p>In the case where a data element is made up of a number of sub elements the SE notation may be used to indicate which sub element should be checked.</p> <p>Some typical examples would be:</p> <p>NET.0?00.DE.004 = The amount field in any network message ending in 00 e.g. this would match with 0100 and 0200 network messages</p> <p>NET.0100.DE.055.TAG.9F36 = The ATC in an authorization request</p> <p>NET.0100.DE.055.TAG.95.BYTE.0-3 = The TVR in an authorization request</p> <p>NET.0100.DE.055.TAG.95.BYTE.1 = The second byte (offset=1) of the TVR (length is assumed to be 1 when not specified)</p> <p>The .BYTE syntax may be further qualified to indicate a particular bit using the syntax .BIT.OFFSET-LEN. The length is assumed to be 1 if not specified.</p> <p>Some examples are:</p> <p>NET.0100.DE.055.TAG.95.BYTE.0.BIT.0 - TVR: No ODA performed</p> <p>NET.0100.DE.055.TAG.95.BYTE.0.BIT.1 - TVR: SDA failed</p> <p><b>Warning:</b> Care needs to be taken since the value specified is a bit offset and not a bit number. Zero represents the first bit found in the data when it is considered to be a binary string. For example, the binary string "10000000" is set to 1 at offset 0.</p> <p>The .TAG syntax may include further child .TAG qualifications in the case where a nested TAG structure is present in the data to be checked.</p> <p><b>Note:</b> The notation for NET data items uses network message identifiers, data elements and tags for ISO 8583 message types. For non-ISO message types it's described in a mapping table how the relevant NET.0?00.DE.???.TAG.???? can be retrieved from the non-ISO message. Whenever necessary the payment system's specific mapping tables are described in the following documents:</p> <table border="1" data-bbox="441 1836 1441 2038"> <thead> <tr> <th data-bbox="441 1836 827 1892">Payment System's name</th><th data-bbox="827 1836 1441 1892">Mapping document reference</th></tr> </thead> <tbody> <tr> <td data-bbox="441 1892 827 1948"></td><td data-bbox="827 1892 1441 1948"></td></tr> <tr> <td data-bbox="441 1948 827 2004"></td><td data-bbox="827 1948 1441 2004"></td></tr> <tr> <td data-bbox="441 2004 827 2047"></td><td data-bbox="827 2004 1441 2047"></td></tr> </tbody> </table>	Payment System's name	Mapping document reference						
Payment System's name	Mapping document reference								

Tag in TSE file	Description
	<p>In the case of APDU data elements the further qualification is as follows:</p> <ul style="list-style-type: none"> <li>• APDU <ul style="list-style-type: none"> <li>• P1 - Represents the P1 data sent to the card by the interface device <ul style="list-style-type: none"> <li>◦ BIT.OFFSET-LEN</li> </ul> </li> <li>• P2 - Represents the P2 data sent to the card by the interface device <ul style="list-style-type: none"> <li>◦ BIT.OFFSET-LEN</li> </ul> </li> <li>• IFD - Represents data sent to the card by the interface device <ul style="list-style-type: none"> <li>◦ BYTE.OFFSET-LEN TAG.TAG_NO</li> <li>◦ TAG.TAG_NO.BYTE.&lt;index&gt; where index is using EMVCo nomenclature</li> </ul> </li> <li>• ICC - Represents data returned by the card in response to the command <ul style="list-style-type: none"> <li>◦ BYTE.OFFSET-LEN</li> <li>◦ TAG.TAG-NO</li> </ul> </li> <li>• SW12 - Represents the status words returned in the APDU response <ul style="list-style-type: none"> <li>◦ BYTE.OFFSET-LEN</li> </ul> </li> </ul> </li> </ul> <p>Some examples are:</p> <p>APDU.00B201.ICC.TAG.70.TAG.61.TAG.50 = Application Label read from record 1</p> <p>APDU.80A8.IFD.BYTE.0-2 = PDOL sent in GPO APDU</p> <p>APDU.80A8.SW12 = Status words returned from GPO command</p> <p>APDU.80AE.IFD.BYTE.0-6 = GEN AC Amount value sent</p> <p>APDU.80AE.P1.BIT.3 = Bit at offset 3 in the GEN_AC P1 parameter</p> <p>APDU.80AE.ICC.TAG.77.TAG.9F10 = Issuer Application data in GEN AC response</p> <p>APDU.80AE.I1.IFD.TAG.95.BYTE.5 = the right most byte or the TVR</p> <p>APDU.80AE.I1.IFD.TAG.95.BYTE.1-2 = the 2 first (left most) bytes of the TVR</p> <p>APDU.80AE.I1.IFD.TAG.95.BYTE.1.BIT.8 = the left most bit of Byte 1</p> <p>In the case of ATR elements the only possible further qualification is .BYTE.OFFSET-LEN to indicate a particular byte of the response:</p> <p>ATR.COLD.BYTE.0-5 = The first 5 bytes of the ATR</p> <p>The data item field may also be in the form LENGTH (&lt;data_item&gt;) to indicate that the test needs to be performed on the length of the data rather than on the data itself. The LENGTH operator is only used on variable length fields.</p> <p>The units of length depends on the format of the field. For example, in the case of numeric fields the length is the number of digits and not the number of bytes since two digits can fit in each byte.</p>

Tag in TSE file	Description
	<p>The data item field may also contain a "known" check rather than a data item definition. These checks typically reflect the operation of the network simulator rather than the value of a data item. The following known checks are defined:</p> <ul style="list-style-type: none"> <li>• SIM_ARQC_VALIDATE - Check that the simulator has validated the ARQC correctly</li> <li>• SIM_PIN_DECRYPTION - Check that the simulator has correctly decrypted the PIN</li> <li>• SIM_PIN_VALIDATION - Check that the simulator has validated the PIN</li> <li>• SIM_TRACK1CVC3_VALIDATE - Check that the simulator has correctly validated the Track1 CVC3 value</li> <li>• SIM_TRACK2CVC3_VALIDATE - Check that the simulator has correctly validated the Track2 CVC3 value</li> <li>• LOG_RESET - Reset log position to the beginning of the log for this test case (see below)</li> </ul> <p>During checking, tools must find the position in the logs where the test cases starts and then interpret any instance parameter relative to this position. For example, a data element defined as NET.0100.DE.055.TAG.95 should cause the tool to start scanning from the start of the log for the test until it finds the first 100 network message.</p> <p>On the other hand a definition starting with NET.0100I2.DE.055.TAG.95 indicates that the tool needs to find the second instance of a NET.0100 message from the start of the log for the test case. In the same way checking tools must also support instance counts when searching for APDUs.</p>
DataFormat	<p>This field describes the expected format of the data item and is comprised of a single letter prefix followed by a length indicator:</p> <ul style="list-style-type: none"> <li>• nL = numeric data of length L digits e.g. "1234" is of format "n4"</li> <li>• hL = hex data of length L nibbles e.g. "FF" is of format "h2"</li> <li>• aL = alphanumeric data of length L characters e.g. "hello" is of format "a5"</li> <li>• bL = binary data of length L bits</li> </ul> <p>Variable length data is supported by appending a following "-" and a maximum length. For example:</p> <p>n1-19 - variable length number between 1 and 19 digits e.g. a PAN</p> <p>h4-8 - A hex value of 2 to 4 bytes (4 to 8 nibbles)</p>
DataItemName	<p>This field contains a more "friendly" human readable name of the data item to be checked. The intention is that this name is used when interacting directly with users rather than the hierarchical name described above.</p>

Tag in TSE file	Description
Operator	<p>This field contains the operator to be used when comparing the data item with the value field. The following operators may be present:</p> <ul style="list-style-type: none"> <li>• "=" - test for equality with the value</li> <li>• "&lt;&gt;" - test for not equals to the value</li> <li>• "&gt;" - test for greater than the value</li> <li>• "&lt;" - test for less than the value</li> <li>• "&lt;=" - test for less than or equals to the value</li> <li>• "&gt;=" - test for greater than or equals to the value</li> <li>• "find" - See below</li> <li>• "exists" - the test passes if the element exists regardless of the value</li> <li>• "not exists" - the test passes if the element doesn't exist</li> <li>• "before" - the test passes if the element appears in the log before another element</li> <li>• "after" - the test passes if the element appears in the log after another element</li> <li>• "not after" - the test passes if the element appears in the log before another element</li> <li>• "like" - the test passes if the element is like the value (see below)</li> </ul> <p>The find operator is used to instruct tools to search forward and find the first instance of a network message or APDU where the data item is equal to the value provided in the value field.</p> <p>For example, a particular test may allow an operator to perform a number of authorization requests using different cards in any order they wish. The "find" operator could be used to set the current position to the correct instance by looking for a PAN with a particular value.</p> <p>The "like" operator is used to compare a data element with a pattern string. At present the pattern string supports two wildcard characters:</p> <ul style="list-style-type: none"> <li>* – matches zero or more characters</li> <li>? – matches any single character</li> </ul> <p>Typical examples include:</p> <pre>NET.0?00.DE.004 like STRING(*30) or NET.0?00.DE.004 like STRING(????30) - Matches an amount (DE04) that ends in 30 NET.0?00.DE.004 like STRING(30*) - Matches an amount that starts with 30</pre>
Value	<p>This field holds the value to be compared with the data item using the defined operator. The field uses one of the following formats:</p> <ul style="list-style-type: none"> <li>• HEX(&lt;hex_value&gt;) - A hexadecimal value e.g. HEX(9000)</li> <li>• NUMBER(&lt;number&gt;) - A numeric value e.g. NUMBER(12345)</li> <li>• BINARY(&lt;binary&gt;) - A binary value e.g. BINARY(101)</li> <li>• STRING("&lt;text&gt;") - A string value e.g. STRING("Mastercard")</li> <li>• ITEM(&lt;data_item&gt;) - A data item e.g. ITEM(APDU.80AE.IFD.BYTE.24-4)</li> <li>• Free Text - A text description of the manual check (only when data item name is MANUAL)</li> </ul>

Tag in TSE file	Description
	<p>The length of the data in the value field is typically padded to match the data format. For example if the format is "h32" then the hex comparison value specified contains 16 bytes (32 nibbles).</p> <p>In the case where the DataItem is using the LENGTH operator the available options are reduced:</p> <ul style="list-style-type: none"> <li>NUMBER (&lt;number&gt;) - A numeric value that indicates the length in units dependant on the data type. For example, for "hL" fields the units are nibbles and for "aL" fields the units are bytes (see section 8.10).</li> <li>LENGTH(&lt;data_item&gt;) - The length of another data item</li> <li>ITEM(&lt;data_item&gt;) - The value of another data item (See note below)</li> </ul> <p>NOTE: In the case where a length is compared directly against another ITEM then the length of the data item should always be compared as a byte length regardless of the data type. For example, consider the following data item definitions:</p> <p>item1 = 0123 item2 = 2</p> <p>The following tests should both evaluate to true:</p> <ul style="list-style-type: none"> <li>LENGTH(item1) = 4 - The length is compared in numeric digits</li> <li>LENGTH(item1) = item2 - The length is compared in bytes</li> </ul>
ValueItemName	In the case where the comparison is with another data item, this field contains the "friendly" name of the other item involved.
ActionIfTrue	<p>This field indicates what action to take if the result of the comparison is true. The field may contain one of two values:</p> <ul style="list-style-type: none"> <li>"PASS" - This indicates that the entire check has passed and there is no need to perform any additional steps for this check. Processing should continue on the check that has the next highest check number.</li> <li>&lt;step no&gt; - This indicates that an additional step needs to be performed before the check can be considered to be complete. The &lt;step no&gt; indicates that control should move to the indicated step number for this check. This approach effectively implements "AND" functionality whereby multiple steps have to succeed for the check to pass.</li> </ul>
ActionIfFalse	<p>This field identifies what action to take if the result of the comparison is false. The field may contain one of two values:</p> <ul style="list-style-type: none"> <li>"FAIL" - This indicates that the entire check has failed and there is no need to perform any additional steps for this check. Processing should continue on the check that has the next highest check number.</li> <li>&lt;step no&gt; - This indicates that an additional step needs to be performed before the check can be considered to have failed. The &lt;step no&gt; indicates that control should move to the indicated step number of this check. This approach effectively implements "OR" functionality where there are alternative ways for the check to pass.</li> </ul>

## **4.4 Test Report**

The machine-readable Test Report is an updated version of the L3 TSE-generated Test Session Files. It provides the test session results, the related logs and tester observations.

The Test Report file is an archive file renamed .tsez and made of:

- The .tse file as described in chapter 4.2,
- The test session logs files made of Card Terminal logs and On-line messages respectively described in chapter 4.6 and 4.7. The test session logs are all located at the archive root.

## 4.5 Machine-readable Test Card Image File Format

This part of the guidelines describes the EMVCo Level 3 Test Card requirements and as a derivative thereof a process for qualifying Test Cards or Card Simulation.

The design principles for the Card Image definitions are:

1. It should be reasonably simple to create and maintain card definitions files manually
2. The card definition should be machine readable to facilitate automated implementation by test tool Vendors,
3. The card definition should unambiguously describe the EMVCo Level 3 required behavior,
4. The card definition should serve as input to the EMVCo Level 3 Card Qualification process. This process then should validate the behavior as defined in the Card image definitions, not less, not more.

**Note:** these guidelines do not prescribe any specific method, process or tool for creating the XML card images. The creation method will solely be at the discretion of each card image file providers.

#### 4.5.1 EMVCo Level 3 Card Application Behavior

The card application that is defined by the card definition file should offer support for:

1. The following EMV Commands as defined in the EMV Specifications version 4.3.
  - a. Compute Cryptographic Checksum (Mastercard)
  - b. External Authenticate
  - c. Generate AC
  - d. Get Challenge
  - e. Get Data
  - f. Get Magstripe Data (JCB Legacy)
  - g. Get Processing Options
  - h. Internal Authenticate
  - i. Issuer Scripts: Application Block, Card Block, PIN Change, PIN Unblock, Put Data
  - j. Read Record
  - k. Select Application
  - l. Verify PIN

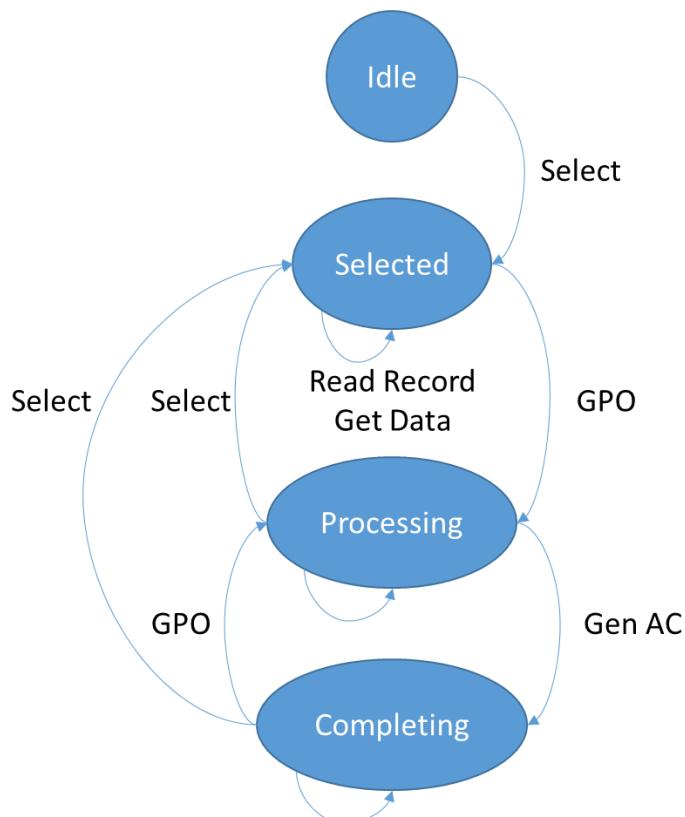
For all other commands issued by the terminal to the card, a SW indicating a failure should be given.

Note that whether or not the card will support/reply successfully to these commands may depend on personalization data and the correctness of the command:

Command	Criteria	Behavior
<b>Compute Cryptographic Checksum</b>	AIP	If card needs to respond, the command should be present in the image definition. If the command is not in the image definition the card should respond with not OK SW, e.g. 6985.
<b>External Authenticate</b>	AIP	Response depending upon the AIP B1b3 indicator.  Please note that if it is important for the Payment System to test that the ExtAuth is successful, then this should be reflected in the pass criteria. It should preferably not be hidden in the card logic to decline in case of an invalid ExtAuth, although the card image definition allows for it in the 2 <sup>nd</sup> Gen AC response.
<b>Get Processing Options, Generate AC</b>	Proper DOL data	These commands will be specified in the image definition as necessary. The correctness of the DOL data as specified by the card are tested by EMVCo Level 2, so no need to check that in Level 3. Implicitly this is however often checked, in particular by the application cryptogram.
<b>Get Challenge, Verify</b>	CVM List	Always respond as if (Enciphered) PIN is supported by the card, as the correctness of the terminal's behavior on CVM lists is covered by EMVCo (test cases 2CJ).
<b>Get Data</b>	See below	
<b>Internal Authenticate</b>	AIP	Response depending upon the AIP B1b6 indicator
<b>Issuer scripts</b>		The default answer of the card will be 9000.

Command	Criteria	Behavior
		In case the test tool implemented the MAC validation, a status word of 6982 is expected at an invalid MAC.
<b>Read Record</b>		In case a non-9000 answer to an issuer script is expected from the card, this needs to be specified in the card image.
<b>Read Record</b>		Only the records specified in the definition should get an answer.

The application should adhere to the following simple state machine:



The following table describes the commands that are allowed in a certain state and what would be the next state after successful completion of the command:

Command / State	Idle	Selected	Processing	Completing
<b>Compute Cryptographic Checksum*</b> <b>(Mastercard)</b>	--	--	Completing	--
<b>External Authenticate*</b>	--	--	--	Completing
<b>Generate AC</b>	--	--	Completing	Completing
<b>Get Challenge</b>	--	--	Processing	--
<b>Get Data</b>	--	Selected	Processing	Completing
<b>Get Magstripe Data* (JCB Legacy)</b>	--	--	Completing	--
<b>Get Processing Options*</b>	--	Processing	--	--
<b>Internal Authenticate*</b>	--	--	Processing	--
<b>Issuer Scripts</b>	--	--	--	Completing
<b>Read Record</b>	--	Selected	Processing	--
<b>Select Application</b>	Selected	Selected	Selected	Selected
<b>Verify PIN</b>	--	--	Processing	--

\* These commands may occur only once during that state.

Next to that, the application should support the following:

1. Correct dynamic ODA handling is required in order to work with real life terminals.
2. Real Application Cryptogram (AC) calculation
3. ARPC validation
4. Compute Cryptographic Checksum (Mastercard), and dynamic Magstripe for other brands.
5. Select Next and Select for blocked application.
6. Get Data at least for the tags in the table below. These are the ones required by EMV. For all other tags a SW indicating a failure, like 6985 is allowed.

Tag	Meaning	Get Data Response	Comment
<b>9F13</b>	Last Online ATC	Return any value below the ATC	
<b>9F17</b>	PIN Try Counter	Return current value	Specify this value in Internal tags to support PIN Try Limit reached test cases.
<b>9F36</b>	ATC	Return current value	
<b>9F4F</b>	Transaction Log Format	9F2701 9F0206 5F2A02 9A03 9F3602 9F5206	In case another response is required, this can be specified in the Internal tags.

The following behavior is NOT mandated by EMVCo Level 3:

1. MAC validation at issuer scripts.
2. Offline Enciphered PIN validation may be required by a Payment System.
3. Card Risk Management functions are not required, as a terminal test process should not be interested in internal card reasoning. Therefore, the CID answered by the Card will be specified in the definition: it either follows the suggestion of the terminal / online authorization or it overrides it with a hardcoded ARQC or an AAC.
4. CVR calculation, as there is no real need from EMVCo Level 3 perspective.<sup>7</sup> Please note, this is not going to say that the CVR should never be populated / fixed. Test tool vendors are free to either compute a correct CVR or take a fixed dummy value. The same holds for Counters and Limits.
5. Previous Transaction History
6. The ATR should be free to choose by the vendors.

Finally, EMVCo Level 3 cards may require special behavior. This might be

1. A fixed value for a piece of data that is normally variable,
2. Padding in records, or an empty record,
3. On device CVM status.

The card definition format will allow for a machine readable way to describe this behavior unambiguously.

---

<sup>7</sup> Note that at some Payment Systems checks the CVR to understand whether the PIN was send to the card and handled correctly. These tests should then be replaced by VERIFY result validations.

#### 4.5.2 EMVCo Level 3 Card Image Definition

The Card Image Definition will be an XML with the following characteristics:

1. The XML should reflect the version of the format spec that it used.
2. Header information is included, such as:
  - a. Card Identifier
  - b. Version Number
  - c. Date Timestamp
  - d. Author
- e. For test plan support it is preferred to have the test card objective / reason for being / special behavior indicated in human readable language along with the card. This can be put in the Description tag in the header.
3. Card Characteristics & Special behavior that cannot be deduced from the person content below, or that should be highlighted for usability reasons. This might include the cryptography used, e.g. M/Chip 4 1.1 / VCPS 2.1 / CPA, etc. However, defining the CVN may be sufficient if not needed by the Payment Systems.
4. Multiple applications on one card are allowed, including (P)PSE.
5. Cryptographic information.

**Note:** Pseudo functions (e.g. emvcard.arqc()) might be used in Card Image Definition. A description of these Pseudo functions is available on request from EMVCo or on the EMVCo web site.

An example of a contact Card Image definition (MP50 v1.1.xml) can be obtained through EMVCo.

To ensure adherence to the format prescribed in this section, EMVCo can provide an XML Schema Definition file (EMVCo L3 Card Definition Format 1.1.xsd). Please contact EMVCo for a copy of this XSD.

## 4.6 Card Terminal Log Format

If submission requires Card Terminal logs that capture transaction details between test cards (or card simulators) and the terminal being tested during terminal integration testing, this section explicitly defines the structure of the Card Terminal log Format.

**Note:** This section does not define how logs should be validated or managed by each entity using EMVCo Level 3 process. This will be at the discretion of each entity to determine.

### 4.6.1 Context

The current POS terminal integration testing uses physical test cards or Card Simulator tools. During testing, the Card - Terminal communication is typically captured using a proprietary log format. This approach has several limitations, among them is that its scalability is limited and it may not be entirely future proof.

This section defines a card-terminal communication log format. It explicitly defines a structure for sending complete contact and contactless Card Terminal logs but does not define how such logs should be tested or handled.

### 4.6.2 Overview

The fundamental encoding of the Card-Terminal Communication Log is XML. Major programming languages of interest to software vendors contain widespread support for XML at this time. XML is intended to be a human readable and position independent format; an XML log allows for tool vendor discretion in utilizing whitespace for their convenience. Unless otherwise specified the ordering of elements is unimportant, whitespace between tags is ignored as is leading and trailing whitespace for a tag's value. Only tags defined by this specification may be used.

The primary purpose of the Card-Terminal Communication Log is data-interchange. In order to map the data onto native programming language structures more easily, attributes which modify the type of an element are foregone. All complex element types will either have required elements and so map onto a structure or have multiple of the same element and so map onto an array or linked list. All other element types are simple.

The basic validity of a Card-Terminal Communication Log is checked with the aid of an XML Schema Definition presented below. The schema denotes as a first basis which fields are mandatory. The tool consuming the logs may enforce non-schema related logical requirements on the content which will be discussed in the following sections.

Each Card-Terminal Communication Log has four mandatory elements contained in the root element <CardTerminalLog>, at its most basic level a log will look like the following:

**Table 4.4: Basic Log**

```
<?xml version="1.0" encoding="utf-8"?>
<CardTerminalLog>
  <LogDetails>...</LogDetails>
  <Contact>...</Contact>
  <Contactless>...</Contactless>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">...<Signature>
</CardTerminalLog>
```

The ellipses (...) indicate that there are sub elements. Hereby a short description of each element:

- The *<LogDetails>* element stores administrative information related to the creation and purpose of the log.
- The *<Contact>* and *<Contactless>* elements store actual communication logs of the same format.
- The *<Signature>* element is encoded according to the XML Digital Signature standard guidelines [3].

These requirements are enforced by the following XML Schema Definition fragment, where the definitions of the individual sub elements will be described more fully in the sections to come:

**Table 4.5: Root Level Schema Fragment**

```
<xss:element name="CardTerminalLog">
  <xss:complexType>
    <xss:all>
      <xss:element name="LogDetails" minOccurs="1" type="logDetails"/>
      <xss:element name="Contact" minOccurs="1" type="interfaceLog"/>
      <xss:element name="Contactless" minOccurs="1"
        type="interfaceLog"/>
      <xss:element ref="ds:Signature" minOccurs="1"/>
    </xss:all>
  </xss:complexType>
</xss:element>
```

#### **4.6.3 Log Details**

Certain administrative information is required by the receiving tool for understanding who created the log and why. This information can be used to associate the log with a particular customer and/or a test case. Certain cataloguing information is also provided for. Each *<LogDetails>* element must contain the following elements:

1. *<Date-Time>* – This tag must contain a standard UTC timestamp in ISO-8601 format. UTC is mandated for orderly coordination of logs arriving from global regions.
2. *<LoggingTool>* – This complex tag contains the name and version of the product producing the logs in the tags *<ProductName>* and *<ProductVersion>* respectively.

3. <*SchemaSelectionIndex*> – For this version of the Card-Terminal Communication Log, this tag must contain the number 1. This field will always contain the major version integer; any change to the schema will result in a new version number, minor changes to this document which do not modify the XML schema will not affect this number.
4. <*Reference*> – This field contains the test case reference number and project information which is generated and provided by the consuming tool.

An example of log details is provided below:

**Table 4.6: Log Details Example**

```

<?xml version="1.0" encoding="utf-8"?>
<CardTerminalLog>
...
<LogDetailsDate-Time>2013-08-21T10:43:46Z</Date-Time>
  <LoggingTool>
    <ProductName>Test Tool Name</ProductName>
    <ProductVersion>1.0</ProductVersion>
  </LoggingTool>
  <SchemaSelectionIndex>1</SchemaSelectionIndex>
  <Reference>SE:502101143;TestCase: XYZ-POS001</Reference>
</LogDetails>
...
</CardTerminalLog>

```

The log details are validated using the following XSD:

**Table 4.7: Log Details Schema Fragment**

```

<xss:complexType name="logDetails">
  <xss:all>
    <xss:element name="LoggingTool" minOccurs="1"
      type="nonEmptyString" />
    <xss:element name="Reference" minOccurs="1" type="xs:string">
      <xss:element name="Date-Time" minOccurs="1" type="dateTimeUTC" />
      <xss:element name="SchemaSelectionIndex" minOccurs="1"
        type="schemaSelectionIndex" />
    </xss:all>
  </xss:complexType>
  <xss:simpleTypename="nonEmptyString">
    <xss:restriction base="xs:string">
      <xss:minLength value="1"/>
    </xss:restriction>
  </xss:simpleType>
  <xss:simpleTypename="dateTimeUTC">
    <xss:restriction base="xs:dateTime">
      <xss:pattern value="[0-9\-\:\:]{19}Z"/>
    </xss:restriction>
  </xss:simpleType>
  <xss:simpleTypename="schemaSelectionIndex">
    <xss:restriction base="xs:integer">
      <xss:enumeration value="1"/>
    </xss:restriction>
  </xss:simpleType>
</xss:complexType>

```

Note that the logging tool shall not be empty and that the date-time format is being restricted to UTC.

#### 4.6.4 Interface Logs

The *<Contact>* and *<Contactless>* elements each contain interface logs. These sections each have two required elements:

1. *<Commands>* – Used to log the actual commands sent using the technology defined by the parent element.
2. *<ConsiderSequences>* – Used to convey information regarding which transactions should be considered part of a test case when additional commands may have been sent by accident or because the terminal supports proprietary features.

Each element may contain multiple sub elements of the same type unless they are empty; an empty *<Commands>* element implies that the particular technology was not used. If either Contact or Contactless transactions were performed then there will be commands present as multiple *<CommandResponse>* elements in the *<Commands>* element. These command-response pair elements must occur in the order they occurred. Each *<CommandResponse>* element contains the following tags:

1. *<Command>* – Either a command APDU or an event literal defined below.
2. *<CommandTimestamp>* – Optional. An integer providing the time in milliseconds at which the command was sent or the event occurred. This time may be relative to system time, UTC or to a timer which is initialized at the beginning of testing, whichever is convenient for the logging tool. CommandTimestamp is not mandatory for Tools that do not have a timer and therefore unable to support time stamps.
3. *<Response>* – The response APDU sent by the card or the response of the card to the event defined by the command. May be empty for some commands.
4. *<ResponseTimestamp>* – Optional. An integer providing the time in milliseconds at which the response was received from the card. Must use the same timer as the command timestamp. This element may be empty for some of the events as described below. ResponseTimestamp is not mandatory for Tools that do not have a timer and therefore unable to support time stamps.

APDUs must be hexadecimal strings. They can contain digits [0-9] or hexadecimal characters

[a-f A-F]. Internal space characters (U+0020) between bytes and nibbles are permitted.

Some events must also be logged such as power on/off, ATRs and card insertion. These events are mandatory to:

1. Ensure that proprietary sessions are handled according to EMV requirements.
2. Ensure that interfaces are switched accordingly.
3. Debug issues related to reader failures.

The following event literals are defined for use in the *<Command>* element and must be logged in a session:

1. **EVENT\_CARD\_INSERTION** – The time at which the Contact card is inserted into the reader or the Contactless card/mobile is presented to the reader. Response and response timestamp must be empty.
2. **EVENT\_CARD\_REMOVAL** – The time at which the card is removed from the reader. Response and response timestamp must be empty.
3. **EVENT\_COLD\_RESET** – The time at which the reader has completed a cold reset of a Contact card or protocol activation of a Contactless card. The response must contain the ATR or ATS if received and the timestamp must be filled accordingly.
4. **EVENT\_POWER\_OFF** – The time at which the card or contactless field is powered off by the reader. Response and response timestamp must be empty.
5. **EVENT\_POWER\_ON** – The time at which the card or contactless field is powered on by the reader. Response and response timestamp must be empty.
6. **EVENT\_WARM\_RESET** – The time at which the reader has completed a cold reset of a Contact card. The response must contain the ATR and the timestamp must be filled accordingly.

The response tag may contain the literal **CARD\_EXCEPTION** if there was an error in the communications with the card where a response is expected.

The *<ConsiderSequences>* element contains optional *<Sequence>* elements. If there were more transactions in a log than are expected or multiple transactions are included out of the order they would otherwise be expected, then these elements may be used to dictate to the receiving tools which sequences of commands should be considered a transaction and in which order they should be considered. The order in which the *<Sequence>* elements occur is therefore crucial and must not be rearranged by receiving tools. Each *<Sequence>* element must contain the following elements in order to define a sequence of commands as a transaction to be considered:

1. *<Start>* – The index of first *<CommandResponse>* sub element in the *<Commands>* element to include in the transaction being defined. Indices begin at 0 in c programming language style.
2. *<End>* – The index of last *<CommandResponse>* sub element in the *<Commands>* element to include in the transaction being defined.

The following is an example of the commands up to SELECT PSE sent during a typical Contact transaction:

**Table 4.8: Example Card Log Section**

```
<?xml version="1.0" encoding="utf-8"?>
<CardTerminalLog>
...
<Contact>
  <ConsiderSequences>
    <Sequence>
      <Start>0</Start>
      <End>20</End>
    </Sequence>
  </ConsiderSequences>
  <Commands>
    <CommandResponse>
      <Command>Power_ON</Command>
      <CommandTimestamp>3789345623</CommandTimestamp>
      <Response/>
      <ResponseTimestamp/>
    </CommandResponse>
    <CommandResponse>
      <Command>COLD_RESET</Command>
      <CommandTimestamp>3789345650</CommandTimestamp>
      <Response>3F6D 00 00</Response>
      <ResponseTimestamp>3789345833</ResponseTimestamp>
    </CommandResponse>
    <CommandResponse>
      <Command>
        00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00
      </Command>
      <CommandTimestamp>3789345913</CommandTimestamp>
      <Response>
        6F 15 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 03
        88 01 01 90 00
      </Response>
      <ResponseTimestamp>3789346735</ResponseTimestamp>
    </CommandResponse>
    ...
  </Commands>
</Contact>
...
</CardTerminalLog>
```

Both the Contact and Contactless sections are validated by the following schema definition:

**Table 4.9: Interface Log Schema Fragment**

```

<xs:complexType name="interfaceLog">
  <xs:all>
    <xs:element name="ConsiderSequences" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Sequence" type="sequence" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Commands" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CommandResponse" type="commandResponse"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>
<xs:complexType name="sequence">
  <xs:all>
    <xs:element name="Start" type="positiveInteger"/>
    <xs:element name="End" type="positiveInteger"/>
  </xs:all>
</xs:complexType>
<xs:simpleType name="positiveInteger">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="commandResponse">
  <xs:all>
    <xs:element name="Command" type="xs:string"/>
    <xs:element name="CommandTimestamp" type="stringInteger"/>
    <xs:element name="Response" type="xs:string"/>
    <xs:element name="ResponseTimestamp" type="stringInteger"/>
  </xs:all>
</xs:complexType>
<xs:simpleType name="stringInteger">
  <xs:restriction base="xs:string">
    <xs:pattern value="([0-9])*/>
    <xs:whiteSpace value="collapse"/>
  </xs:restriction>
</xs:simpleType>
```

#### 4.6.5 Signature

In order to verify the integrity of the log and to ensure that an authentic Payment system-qualified tool was used to generate the log, an XML Digital Signature must be attached to each log. All signatures will therefore be enveloped.

A receiving tool must be provided with the RSA public keys in a standard format each associated with an accompanying unique name which will be associated with a given tool as identified by its <LoggingTool> element.

The <KeyName> tag must be used by the sending tool and must contain one of the unique names of the keys provided beforehand to indicate the key which was used to generate the signature and to enable the receiving tool to check its key stores to establish trust in the key.

Only canonicalization without comments is supported and only the enveloped signature transform is supported.

#### 4.6.6 Schema and Validation

Each file received will go through the following validations:

1. Check the XML document to ensure it is well-formed.
2. Check the XML against the log format schema.
3. Check that the key used to generate the signature is trusted.
4. Check that the signature is correct.

Failures in any of these validations will cause rejection of the log without response. These checks should therefore be done by the sending party prior to upload.

These requirements take precedence over any other requirements described above.

Several online tools are available for checking XSD validity, XML well-formed and to validate XML files against to a particular XSD. The following URL provides an example of utilities for schema validation <http://www.utilities-online.info/xsdvalidation>.

Developers are free of using any of those tools to ensure a proper card terminal log schema validation format during development.

The complete schema is shown below in textual form.

**Table 4.10: Complete Log Schema**

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
        20020212/xmldsig-core-schema.xsd"/>
    <xs:element name="CardTerminalLog">
        <xs:complexType>
            <xs:all>
                <xs:element name="LogDetails" minOccurs="1"
                    type="logDetails"/>
                <xs:element name="Contact" minOccurs="1" type="interfaceLog"/>
                <xs:element name="Contactless" minOccurs="1"
                    type="interfaceLog"/>
                <xs:elementref name="Signature" type="ds:Signature" minOccurs="1"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
<xs:complexType name="logDetails">
```

```

<xs:all>
  <xs:element name="LoggingTool" minOccurs="1"
  type="loggingTool"/>
    <xs:element name="Reference" minOccurs="1" type="xs:string"/>
    <xs:element name="Date-Time" minOccurs="1" type="dateTimeUTC"/>
    <xs:element name="SchemaSelectionIndex" minOccurs="1"
  type="schemaSelectionIndex"/>
</xs:all>
</xs:complexType>
<xs:complexType name="loggingTool">
  <xs:all>
    <xs:element name="ProductName" minOccurs="1"
  type="nonEmptyString"/>
    <xs:element name="ProductVersion" minOccurs="1"
  type="nonEmptyString"/>
  </xs:all>
</xs:complexType>
<xs:simpleType name="nonEmptyString">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="dateTimeUTC">
  <xs:restriction base="xs:dateTime">
    <xs:pattern value="[0-9\-\:\:]{{19}}Z"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="schemaSelectionIndex">
  <xs:restriction base="xs:integer">
    <xs:enumeration value="1"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="interfaceLog">
  <xs:all>
    <xs:element name="ConsiderSequences" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Sequence" type="sequence" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Commands" minOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CommandResponse" type="commandResponse"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>
<xs:complexType name="sequence">
  <xs:all>
    <xs:element name="Start" type="positiveInteger"/>
    <xs:element name="End" type="positiveInteger"/>
  </xs:all>
</xs:complexType>
<xs:simpleType name="positiveInteger">
  <xs:restriction base="xs:integer">

```

```

<xs:minInclusive value="0"/>
</xs:restriction>
</xs:simpleType>
<xs:complexType name="commandResponse">
<xs:all>
<xs:element name="Command" type="xs:string"/>
<xs:element name="CommandTimestamp" type="xs:integer"/>
<xs:element name="Response" type="xs:string"/>
<xs:element name="ResponseTimestamp" type="stringInteger"/>
</xs:all>
</xs:complexType>
<xs:simpleType name="stringInteger">
<xs:restriction base="xs:string">
<xs:pattern value="([0-9])*/>
<xs:whiteSpace value="collapse"/>
</xs:restriction>
</xs:simpleType>
</xs:schema>

```

#### 4.6.7 Examples

The examples provided below illustrate typical outputs of a test tool creating transaction logs based on these specifications.

**Note:**

- *Whitespaces have been added to these illustrations for presentation purposes only.*
- *These examples are provided with a public key for signature validation purposes. The actual key exchange procedure is described above in the ‘Signature’ section.*

**Table 4.11: Simple Log – illustrating use of signature**

```

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<CardTerminalLog>
<LogDetails>
<LoggingTool>
<ProductName>Card Simulator Test Tool</ProductName>
<ProductVersion>5.0</ProductVersion>
</LoggingTool>
<Reference/>
<Date-Time>2013-08-19T14:32:00Z</Date-Time>
<SchemaSelectionIndex>1</SchemaSelectionIndex>
</LogDetails>
<Contact>
<ConsiderSequences/>
<Commands/>
</Contact>
<Contactless>
<ConsiderSequences/>
<Commands/>
</Contactless>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>

```

```

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>mFwUrHk1SDj1x0WgQFITY84=</DigestValue>
</Reference>
</SignedInfo>

<SignatureValue>LEywXQ9uCKffO9TGwSoGyZ+cjOiBUADTZ6UzRn/W zum Q64KlxhcX Yom J
4+65AcVYyW7PAj/kpPmBaa4eTbuneZChIVZMdY4fJvtX8pxE5rVMij9WkhoFxHKYd5VJaMUE
hoXny8/iNtZmsqlQZ8OLwmMuheo8+r05ck9nEX0N/G0=</SignatureValue>
<KeyInfo>
  <KeyName>Card Simulator RSA Log Key 1</KeyName>
</KeyInfo>
</Signature>
</CardTerminalLog>

```

**Table 4.12: Contact Online Transaction**

```

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<CardTerminalLog>
  <LogDetails>
    <LoggingTool>
      <ProductName>Card Simulator Test Tool</ProductName>
      <ProductVersion>5.0</ProductVersion>
    </LoggingTool>
    <Reference/>
    <Date-Time>2013-08-19T14:32:00Z</Date-Time>
    <SchemaSelectionIndex> 1</SchemaSelectionIndex>
  </LogDetails>
  <Contact>
    <ConsiderSequences/>
    <Commands>
      <CommandResponse>
        <Command>00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31
00</Command>
        <CommandTimestamp>37648753</CommandTimestamp>
        <Response>6F 15 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 03 88 01 01 90
00</Response>
        <ResponseTimestamp>37648753</ResponseTimestamp>
      </CommandResponse>
      <CommandResponse>
        <Command>00 B2 01 0C 00</Command>
        <CommandTimestamp>37648873</CommandTimestamp>
        <Response>70 21 61 1F 4F 08 A0 00 00 00 25 01 04 03 50 10 50 65 72 73 6F 6E 61 6C
20 41 63 63 6F 75 6E 74 87 01 01 90 00</Response>
        <ResponseTimestamp>37648873</ResponseTimestamp>
      </CommandResponse>
      <CommandResponse>
        <Command>00 B2 02 0C 00</Command>
        <CommandTimestamp>37648995</CommandTimestamp>
        <Response>70 20 61 1E 4F 07 A0 00 00 00 29 10 10 50 10 50 65 72 73 6F 6E 61 6C
20 41 63 63 6F 75 6E 74 87 01 02 90 00</Response>
        <ResponseTimestamp>37648995</ResponseTimestamp>
      </CommandResponse>
    </Commands>
  </Contact>
</CardTerminalLog>

```

```

<CommandResponse>
<Command>00 B2 03 0C 00</Command>
<CommandTimestamp>37649094</CommandTimestamp>
<Response>6A 83</Response>
<ResponseTimestamp>37649094</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 A4 04 00 08 A0 00 00 00 25 01 04 03 00</Command>
<CommandTimestamp>37658141</CommandTimestamp>
<Response>6F 2C 84 08 A0 00 00 00 25 01 04 03 A5 20 50 10 41 6D 65 72 69 63 61
6E 20 45 78 70 72 65 73 73 9F 38 03 9F 35 01 87 01 01 5F 2D 02 65 6E 90 00</Response>
<ResponseTimestamp>37658141</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>80 A8 00 00 03 83 01 12 00</Command>
<CommandTimestamp>37658295</CommandTimestamp>
<Response>80 0E 7C 00 08 01 02 00 08 03 03 01 08 04 05 00 90 00</Response>
<ResponseTimestamp>37658295</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 B2 01 0C 00</Command>
<CommandTimestamp>37659789</CommandTimestamp>
<Response>70 81 E2 5F 20 1A 45 58 50 52 45 53 53 50 41 59 20 32 20 43 41 52 44 20
44 55 41 30 32 41 20 20 57 13 37 42 45 45 54 00 00 1D 15 07 20 11 10 71 23 45 00 00 0F
8F 01 97 90 81 90 D0 F0 D4 53 53 02 34 66 4A 8C 33 51 DA 2A 81 08 31 D4 DA EC 6C C1
EA F6 5F 17 B2 C9 8F F0 D5 9B 00 42 B7 65 3E 02 15 99 26 8E 5E 68 A9 A6 FB B0 C1 7E
09 D3 E1 6F CE FD E8 79 55 64 14 C1 7B 2F 84 68 7B 6A 5C A4 4F AD 05 CA 84 7F A8 07
5E 43 5F E2 53 8A 4E 91 56 B6 AD 16 F7 4E E4 0D 4A 2F 5D 88 94 74 31 93 46 CA 59 CD
B6 72 9F B4 AF AF 27 5F D9 A5 22 A6 8A 9D 58 D8 4C 4C F6 CE EB B2 57 EF 5C F6 6C
E1 1F 5C 06 2B 66 61 47 5F 42 2D 92 14 3F 17 C3 B6 70 B7 1A FA 9B 3D EC B5 5D E0 26
FA 4E 62 76 0D 9F 32 01 03 90 00</Response>
<ResponseTimestamp>37659789</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 B2 02 0C 00</Command>
<CommandTimestamp>37660112</CommandTimestamp>
9F 37 04 8D 17 8A 02 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 93
81 80 9B A4 9C 61 20 C8 1B 83 C4 D3 D5 10 B8 D8 22 2E 32 D3 16 65 E7 59 54 DB 8F 98
7B F9 7E 06 01 20 B9 FF 3F 6E 25 67 9E 0D AE 21 95 8C 84 6B B5 7F C3 20 3D F3 F1 DC
34 AA BE A2 74 77 0A 6D BE AA 35 8B B6 DD 2C 9A 7C 9D 67 47 E6 66 FF 84 74 96 48 C9
C8 3A FC 4D 12 43 86 18 24 B8 72 22 F8 96 84 FA E2 A2 E9 C6 8E 4C 34 61 D6 F8 0D 76
B6 AA 58 75 66 7E 99 AB 36 CF A6 3F E7 91 36 63 4E EB 9F 08 02 00 01 9F 07 02 FF 00
5F 28 02 08 26 9F 42 02 08 26 5F 30 02 02 01 9F 4A 01 82 8E 14 00 00 00 00 00 00 00 00 00
42 01 44 03 41 03 5E 03 1F 02 00 00 90 00</Response>
<ResponseTimestamp>37660112</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 B2 03 0C 00</Command>
<CommandTimestamp>37660444</CommandTimestamp>
<Response>70 32 5F 25 03 11 07 01 5F 24 03 15 07 31 5A 08 37 42 45 45 54 00 00 1F
5F 34 01 00 9F 0D 05 F4 70 C4 98 00 9F 0E 05 00 00 00 00 00 9F 0F 05 F4 70 C4 98 00 90
00</Response>
<ResponseTimestamp>37660444</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 B2 04 0C 00</Command>
<CommandTimestamp>37660696</CommandTimestamp>
<Response>70 81 B0 9F 46 81 80 51 D3 C6 38 BC 3F 38 7D F7 D6 15 DB 16 92 61 9A
58 E5 99 BF 3C 53 77 14 51 02 1B 4F 53 1F F2 D5 1D AA 32 2C 8F F2 FB 9C 8B 9C C6 5F
E8 9B 04 92 A7 DA 90 65 47 BB 61 59 C6 55 DC B2 90 76 F7 4F 72 36 56 DE A6 42 B7 B7

```

```

75 93 23 9A 9D 7A 12 7A 94 B8 78 B5 FF C8 82 C0 79 59 02 09 B4 F1 D6 2F 75 34 FF 57
5C 1E FF 7F 08 20 7D 09 2C BB 01 C1 4E 46 68 66 38 A6 4B AC D5 DF 97 C4 D8 94 F2 E3
9F 47 01 03 9F 48 1A A4 52 F3 66 24 FA D9 24 A8 36 60 16 13 7B 09 48 34 DE 15 37 98 85
D5 8D C9 CF 9F 49 08 9F 1A 02 9A 03 9F 37 04 90 00</Response>
<ResponseTimestamp>37660696</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 B2 05 0C 00</Command>
<CommandTimestamp>37660967</CommandTimestamp>
<Response>70 81 A5 9F 2D 81 80 09 A4 0D 46 3C C1 BE 06 C3 FF 3C 6E 67 01 CB
E7 22 20 68 ED 52 AC 4D 61 5B 0A CD 1F 26 52 B7 FB E4 F7 A9 B9 F3 86 5B 7F 5D 3D 1A
12 0B 95 81 60 48 48 C7 AC CC 28 E8 19 9F 3C 6E EE 56 87 0F FE 16 A6 1F D4 6B 53 5C
44 44 26 74 2A 8F 00 05 40 4D 47 02 77 DA 5B 39 F7 46 F5 55 5C 0D 1D 3E 6E 12 4E B8
DF D5 40 6F CD 0B 2F D9 D2 68 AF DB 27 C9 E8 E3 49 26 75 D4 B1 D0 22 80 B2 CF 92 82

D1 9F 2E 01 03 9F 2F 1A 8A 3D 33 64 AE 21 19 87 3E 38 07 A3 9E 41 F8 BE 68 CD F7 94

56 2B 1C 01 F3 4F 90 00</Response>
<ResponseTimestamp>37660967</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>80 AE 80 00 1D 00 00 00 00 01 00 00 00 00 00 00 00 03 92 00 00 00 00 00
03 92 01 01 01 00 01 02 03 04 00</Command>
<CommandTimestamp>37661517</CommandTimestamp>
<Response>80 12 80 01 26 8E BE 59 00 73 B2 F4 4A 06 02 01 03 A0 00 00 90
00</Response>
<ResponseTimestamp>37661517</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>00 82 00 00 0A F4 29 7F 52 45 06 0E 73 30 30 00</Command>
<CommandTimestamp>37661737</CommandTimestamp>
<Response>90 00</Response>
<ResponseTimestamp>37661737</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
<Command>80 AE 40 00 1F 30 30 00 00 00 00 01 00 00 00 00 00 00 03 92 00 00 00
00 00 03 92 01 01 00 01 02 03 04 00</Command>
<CommandTimestamp>37661908</CommandTimestamp>
<Response>80 12 40 01 26 FC C9 5A A7 18 B1 6F 80 06 02 01 03 60 00 00 90
00</Response>
<ResponseTimestamp>37661908</ResponseTimestamp>
</CommandResponse>
</Commands>
</Contact>
<Contactless>
<ConsiderSequences/>
<Commands/>
</Contactless>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>vcr77IJIQPqAdCS3atRfyKvYnfQ=</DigestValue>
</Reference>
</SignedInfo>
```

```

<SignatureValue>y7hDPNYWmX+cMG9qW+HQB2CiU5HiZOT1eyDjLlx1/NIF2KDG2eqhssp
AUUpme+Hs9/57/FhJzyOaWD1UA3lvPORucJW Y/Txo+lwXQN9VK/caQtr7QW
KXPozLbkD1nT JusMGCf9b22e
yse56MjG7J7HhHsbTcT5QJCTubtiDGEI8c=</SignatureValue>
<KeyInfo>
  <KeyName>Card Simulator RSA Log Key 1</KeyName>
</KeyInfo>
</Signature>
</CardTerminalLog>

```

**Table 4.13: Contactless Online Transaction**

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CardTerminalLog>
  <LogDetails>
    <LoggingTool>
      <ProductName>Card Simulator Test Tool</ProductName>
      <ProductVersion>0.0.0</ProductVersion>
    </LoggingTool>
    <Reference>EMVMode 13 1 1 ProjectId 555-1234</Reference>
    <Date-Time>2013-09-13T01:33:38Z</Date-Time>
    <SchemaSelectionIndex>1</SchemaSelectionIndex>
  </LogDetails>
  <Contact>
    <ConsiderSequences/>
    <Commands/>
  </Contact>
  <Contactless>
    <ConsiderSequences/>
    <Commands>
      <CommandResponse>
        <Command>EVENT_POWER_OFF</Command>
        <CommandTimestamp>1379079218857</CommandTimestamp>
        <Response>null</Response>
        <ResponseTimestamp>0</ResponseTimestamp>
      </CommandResponse>
      <CommandResponse>
        <Command>EVENT_POWER_ON</Command>
        <CommandTimestamp>1379079231347</CommandTimestamp>
        <Response>null</Response>
        <ResponseTimestamp>0</ResponseTimestamp>
      </CommandResponse>
      <CommandResponse>
        <Command>00a4 04 00 0e 32 50 41 59 2e 53 59 53 2e 44 44 46
30 31 00</Command>
        <CommandTimestamp>1379079231431</CommandTimestamp>
        <Response>6f3b 84 0e 32 50 41 59 2e 53 59 53 2e 44 44 46 30
31 a5 29 bf 0c 26 61 24 4f 08 a0 00 00 00 25 01 04 03 50 10 41 6d 65
72 69 63 61 6e 20 45 78 70 72 65 73 73 87 01 01 9f 28 02 40 04 90
00</Response>
        <ResponseTimestamp>1379079231624</ResponseTimestamp>
      </CommandResponse>
      <CommandResponse>
        <Command>00a4 04 00 08 a0 00 00 00 25 01 04 03 00</Command>
        <CommandTimestamp>1379079231635</CommandTimestamp>
        <Response>6f2c 84 08 a0 00 00 00 25 01 04 03 a5 20 50 10 41
6d 65 72 69 63 61 6e 20 45 78 70 72 65 73 73 9f 38 03 9f 35 01 87 01
01 5f 2d 02 65 6e 90 00</Response>
      </CommandResponse>
    </Commands>
  </Contactless>
</CardTerminalLog>

```

```

<ResponseTimestamp>1379079231788</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>80a8 00 00 03 83 01 a2 00</Command>
    <CommandTimestamp>1379079231800</CommandTimestamp>
    <Response>8016 5d 80 08 01 01 00 08 02 02 01 10 01 01 00 18
01 01 00 20 01 01 00 90 00</Response>
    <ResponseTimestamp>1379079232070</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>00b2 01 0c 00</Command>
    <CommandTimestamp>1379079232079</CommandTimestamp>
<Response>7076 5f 20 1a 58 50 20 43 41 52 44 20 30 31 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 57 13 37 42 45 45 54 00 00
1d 16 11 70 21 10 70 00 00 00 00 0f 8c 15 9f 02 06 9f 03 06 9f 1a 02
95 05 5f 2a 02 9a 03 9c 01 9f 37 04 8d 17 8a 02 9f 02 06 9f 03 06 9f
1a 02 95 05 5f 2a 02 9a 03 9c 01 9f 37 04 9f 08 02 00 01 9f 07 02 ff
00 9f 42 02 08 26 5f 30 02 07 02 90 00</Response>
    <ResponseTimestamp>1379079232402</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>00b2 02 0c 00</Command>
    <CommandTimestamp>1379079232413</CommandTimestamp>
<Response>7042 5f 25 03 12 11 01 5f 24 03 16 11 30 5a 08 37
42 45 45 54 00 00 1f 5f 34 01 00 9f 0d 05 f4 70 c4 98 00 9f 0e 05 00
00 00 00 00 9f 0f 05 f4 70 c4 98 00 8e 0e 00 00 00 00 00 00 00 42
01 5e 03 1f 02 90 00</Response>
    <ResponseTimestamp>1379079232582</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>80ae 50 00 1d 00 00 00 06 01 00 00 00 00 00 00
08 26 00 00 00 00 08 26 13 09 13 00 00 50 a1 2c 00</Command>
    <CommandTimestamp>1379079232593</CommandTimestamp>
<Response>7781 a7 9f 27 01 40 9f 36 02 00 01 9f 4b 81 90 85 db 8b f4 c1 d0 1a 6f 8b a0 04 be
18 76 a1 91 ae 32 a8 e7 17 0d ea 5f
07 17 c6 08 d3 0b 73 b3 48 53 f6 bc 26 68 57 d4 39 6c b9 2f 50 e0 a8
1b cc 15 6a 15 b7 9f 4d d5 f4 41 02 fd e2 2a fb 03 3c 47 0b 25 46 ad
ba d8 51 a5 0a 3a 64 88 78 c9 57 6e e0 bb f7 a0 3a 5d 6c b3 b1 c4 f9
d8 fc bd f4 69 18 80 e9 24 ec 17 7f 25 6c 1b f4 7d e6 79 aa 69 a0 bb
70 0e a8 97 c9 5f 28 4f da 4e fe 3f 42 66 54 bc 33 0b 61 da 0a fa ea cf 36 69 f4 2d 9f 10 07 06
02 01 03 90 10 00 90 00</Response>
    <ResponseTimestamp>1379079233340</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>00b2 01 14 00</Command>
    <CommandTimestamp>1379079233350</CommandTimestamp>
<Response>7081 c4 8f 01 c7 90 81 90 56 d0 9f 2b 71 67 85 67
db 2b c0 93 52 9d 2e e6 1d ff 8d 28 5d 3b 03 f3 97 07 04 69 77 de 12
75 6c 9a fa 5c 16 8f c1 90 da db 2e 09 48 17 a5 2f 51 ad 75 84 dc 26
46 c7 4a 1e ee 94 3d a6 af 8b eb 4e b9 a6 b9 a3 a2 2e d7 4d db cf bc
92 b1 f5 7f e8 41 e2 68 82 40 91 d3 14 db 18 de 95 4c 81 45 3e f6 dc
11 7c 46 e2 ec 39 5d 39 9d 9c 85 23 c5 c9 f1 2c 2d c4 f3 e7 28 01 69
7e de 89 91 38 8c 76 1b e2 40 5e 2f 54 7e fe 79 b7 17 06 82 ed 92 24
8b a0 5b 9d b9 05 c2 09 08 fa 66 85 a1 bf 6f 2d 93 57 0f b4 ac eb 08
81 4d 17 04 7c b9 e2 0a 82 de 6e 4e 31 9f 4a 01 82 9f 32 01 03 90
00</Response>
    <ResponseTimestamp>1379079233444</ResponseTimestamp>
</CommandResponse>
<CommandResponse>
    <Command>00b2 01 24 00</Command>

```

```

<CommandTimestamp>1379079233455</CommandTimestamp>
<Response>7081 c5 9f 46 81 90 38 ee f0 2c 94 d1 fe 77 ab 93
    7f 06 34 41 a6 db ec ad df 0c 56 58 69 ef da ef 40 a6 06 a0 9c 77 27
    3b 51 a5 63 79 88 50 af ec 45 69 8a bd f7 5d 32 4f 7d 9e 70 9c 89 2f f1 d8 19 44 6a 10 97 68 de
        86 bc cd 8e 43 b8 33 6f a1 e4 92 dc 7c 8f
        11 5d 20 c8 e3 9e 88 b6 dd 2a 36 66 3e ed 25 22 0b dc f3 23 96 6b 59
        34 a7 77 68 9a ea 89 3d ce 1d 56 71 12 6f ae 7e 29 87 91 6d 41 a4 45
        1a c7 37 ea 21 23 37 44 ff 82 97 0c 0f c4 2d 84 32 30 25 9f 47 01 03
    9f 48 2a 80 65 fa 9d 9f ce ec 34 c5 dc 86 65 77 d4 fa 2c e4 d0 49 6c
    16 a0 1e c9 35 80 33 f2 b5 22 ce f3 42 04 57 5a 4f 1a 78 d1 8c 33 90
    00</Response>
        <ResponseTimestamp>1379079233538</ResponseTimestamp>
    </CommandResponse>
    <CommandResponse>
        <Command>EVENT_POWER_OFF</Command>
        <CommandTimestamp>1379079234466</CommandTimestamp>
        <Response>null</Response>
        <ResponseTimestamp>0</ResponseTimestamp>
    </CommandResponse>
</Commands>
</Contactless>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
        <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <Reference URI="">
                <Transforms>
                    <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                </Transforms>
                <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <DigestValue>tp4w4QGL9e0aFBORXAM4I4OQWnI=</DigestValue>
                </Reference>
            <SignedInfo>
                <SignatureValue>WBD1i1REbPmaqykvB5ealvZhBit902cHKxI4SY0gfMcGMIxV1Yq3m
wxNG1ct1Bt8niKc7VNsyit6/pbtJOkIrsBUcKuDiWgHrSWmJCBDDQQ8tZwafKoLYbZK5A
V3WLquqHrX+iO6KeeFkTHVmRUIXGbO1vXAyZ3bWHvG5in9O4=</SignatureValue>
                <KeyInfo>
                    <KeyName>Card Simulator RSA Log Key 1</KeyName>
                </KeyInfo>
            </SignedInfo>
        </Signature>
    </CardTerminalLog>

```

## 4.7 Common Online Message Format

The present section proposes a common format for Online Message log transmitted from Acceptance Device to System, or System to Acceptance Device, or any Device to any other Device). Mainly, this online message format log provides authorisation request and response details between Acceptance Devices under test and the online System, this section explicitly defines the structure of the Online Message Format.

**Note:** This section does not define how message should be managed by each entity using EMVCo Level 3 process. This will be at its own discretion to determine.

#### 4.7.1 Introduction

The EMVCo Level 3 Online Message Format is an XML-based file format which can be used to represent online messages delivered from one payment-related device or system to another payment-related device or system, while supporting the following goals:

- Automation – The format should allow the content of online messages to be tested.
- Vendor and scheme-independence – the format and its IP is owned by the industry.
- Completeness – The format can express the principal online message formats supported by all EMVCo members, as well as other formats used by other organizations.
- Simplicity – The format is easy to implement and provides human-readable and machine-readable data.
- Industry-Standard Fields – The use of which can facilitate code which tests online messages to run, unchanged, across multiple online message formats.

The abbreviated name of the EMVCo Level 3 Online Message Format is EMVCoL3-OMF.

The encoding of the Online Message is XML. Major programming languages of interest to software vendors contain widespread support for XML at this time. XML is intended to be a human readable and position independent format; an XML message allows for tool vendor discretion in utilizing whitespace for their convenience. Unless otherwise specified the ordering of elements is unimportant, whitespace between tags is ignored as is leading and trailing whitespace for a tag's value. Only tags defined by this specification may be used.

The basic validity of an Online Message is checked with the aid of an XML Schema Definition presented below. The schema denotes as a first basis which fields are mandatory. The tool may enforce non-schema related logical requirements on the content which will be discussed in the following sections.

#### 4.7.2 Format specification

An EMVCoL3-OMF file is an XML file which contains the following basic XML blocks:

- Log Details – specifies the application which create the EMVCoL3-OMF file and some other house-keeping information.
- Connection List – specifies a list of data connections which are used during the message exchanges contained within the file. EMVCoL3-OMF supports the inclusion of multiple connections to allow for recording traffic on multiple interfaces.

- Message List – the list of messages recorded across one or more interfaces. Note that EMVCoL3-OMF does not require that all messages passing over a specified connection should be recorded. While EMVCoL3-OMF does support the breakdown of raw messages into individual fields and subfields, the breakdown is not mandatory. The format also supports the arbitrary inclusion of the ToolComment tag to indicate tool-specified comments, such as links becoming active and inactive, processing messages and general comments of any kind.
- Signature – a signature block which digitally signs the contents of the EMVCoL3-OMF file.

An XML Schema Definition (XSD) file is available from EMVCo which can be used to validate the major components of an EMVCoL3-OMF file.

**Table 4.14: Online message format specification**

Field	Block/Tag/Attribute	M/O/C	Description
<EMVCoL3OnlineMessageFormat>	Block-start	M	XML root node.
<LogDetails>	Block-start	M	Block which specifies the application which created the online message file, the file format version and comments.
Date-Time	Tag	M	The date and time when the file was generated. Expressed in UTC.
<LoggingTool>	Block	M	Block which specifies the application which created the online message log.
ProductName	Tag	M	The name of the application which created the online message log.
ProductVersion	Tag	M	The version of the application which created the online message log.
</LoggingTool>	Block-end	M	
SchemaSelectionIndex	Tag	M	The version number of the online message log format used. The version of the standard documented here is “1.0”.
Reference	Tag	M	Any useful reference information for the data contained within the file.
</LogDetails>	Block-end	M	
<ConnectionList>	Block-start	M	The list of connections, each one contained within its own Connection block, which are used during the message exchanges contained within the log. Note that there may be multiple sources and targets which trace online messages as they move from one system to another.
<Connection>	Block-start	M	Block which specifies information concerning one end of a point-to-point connection.
ID	Attribute of <Connection>	M	A unique symbolic name used to refer to this connection.
<Protocol>	Block-start	M	Block which specifies the protocol operating over the connection.
FriendlyName	Tag	M	The user-viewable name of the protocol used on this connection

Field	Block/Tag/Attribute	M/O/C	Description
SymbolicName	Tag	M	<p>The machine-readable name of the protocol used on this connection. The protocol format is made of the PS name and the protocol name separated by a dot</p> <p>Valid values for the PS name are:</p> <ul style="list-style-type: none"> <li>AMEX for American Express</li> <li>DISC for Discover</li> <li>JCB for JCB</li> <li>MC for MasterCard</li> <li>VISA for Visa</li> <li>UP for UnionPay</li> </ul> <p>. Examples:</p> <ul style="list-style-type: none"> <li>• VISA.BASE1</li> <li>• MC.GCIS</li> </ul>
VersionInfo	Tag	M	A freeform, user-viewable text field which specifies the version of the protocol, including compliance, regional variations etc, used on this connection. The information specified in this tag should be sufficient for a knowledgeable reader to be able to recreate the general configuration of the protocol used over the connection.
</Protocol>	Block-end	M	
<TCPIPParameters>	Block-start	C	Block which specifies the details of a TCP/IP connection in sufficient detail that a knowledgeable user is able to recreate the connection.
Address	Tag	M	For a client TCP/IP connection, this tag specifies the address of the remote server to which the client connects. The address value may be specified as a fully-qualified domain name or a dotted quad address. For a server connection, this tag can be left blank.
Port	Tag	M	For a client TCP/IP connection, this tag specifies the port number on the remote server to which the client connects. For a server connection, this tag specifies the port number upon which the server is listening.
Header	Tag	M	A free-form text description of the format of the header bytes, if any, which indicate the size of the upcoming TCP/IP data block – eg “Two bytes, network order” or “Four bytes, PDP-order”.

Field	Block/Tag/Attribute	M/O/C	Description
Client	Tag	M	Specifies whether the connection is operating as a client (true) or a server (false).
Format	Tag	M	A free-format text string which describes the basic format of the data, eg, "ASCII", "EBCDIC", "BCD" etc.
</TCPIPParameters>	Block-end	C	
</DialupParameters>	Block-start	C	Block which specifies the details of a PSTN connection in sufficient detail that a knowledgeable user is able to recreate the connection.
PhoneNumber	Tag	M	For a client connection, this tag specifies the telephone number of the remote server which this connection must dial in order to initiate a call. For a server connection, this tag can remain blank.
ModemInitString	Tag	M	The modem initialization string delivered to the modem when the application starts and which is intended to set the modem to a known functional state sufficient for a call to be initiated.
Client	Tag	M	Specifies whether the connection is operating as a client (true) or a server (false).
Format	Tag	M	A free-format text string which describes the basic format of the data, eg, "ASCII", "EBCDIC", "BCD" etc.
</ConnectionParameters>	Block-end	C	
</Connection>	Block-end	M	
</ConnectionList>	Block-end	M	
<OnlineMessageList>	Block-start	M	Zero or more tool-generated comments and zero or more online messages captured during the logging session. ToolComment tags and OnlineMessage blocks can appear in any order, any position and any number within the OnlineMessageList block.
ToolComment	Tag	O	An application-generated comment concerning any information not explicitly related to a specific online message or data exchange (eg, comments could document connections going up and down, out-of-band keep-alive packets and informational messages of any kind). This tag can appear zero or more times within the enclosing OnlineMessageList block, at any location immediately within it.
<OnlineMessage>	Block-start	O	XML block which specifies a single online message or data exchange from one point to another
Class	Attribute of <OnlineMessage>	M	Symbolic name of the class of message represented by the block. This name has meaning only within the context of the protocol specified in the appropriate Connection block.

Field	Block/Tag/Attribute	M/O/C	Description
Source	Attribute of <OnlineMessage>	M	Symbolic name of originator of the online message or data block. This symbolic name is uniquely specified by the ID attribute of a defined Connection block.
Destination	Attribute of <OnlineMessage>	M	Symbolic name of destination for the online message or data block. This symbolic name is uniquely specified by the ID attribute of a defined Connection block.
RawData	Tag, human-readable binary data	M	The raw, uninterpreted, hexadecimal data bytes which make up the complete online message, space-separated – eg “01 F0 F0 30 31”. Transport-level header bytes (ie, those specified in the Header tag of the ConnectionParameters block) are not included.
MessageGroup	Tag	O	A GUID which can be used to link together related OnlineMessage entries – for example, a related request and response, or a request, repeat-request, response, reversal request, reversal response and so on. If messages are not grouped, then this tag can remain blank.
<MessageInfo>	Block-start	M	Block which specifies a range of processing parameters, functions, features and results which are not explicitly available from the online message itself.
PINValue	Tag	O	The value of the PIN recovered from the online message. This tag should not be present if no PIN was recovered.
PINKey	Tag	O	The value of the PIN key used to encipher and decipher the PIN block. This tag should not be present if no PIN was presented in the online message.
PINValidated	Tag	O	Tag which specifies true, false or N/A to indicate whether the online PIN specified in the online message was, respectively, validated, failed to validate or if no validation of the PIN block was attempted.
ARQCValidated	Tag	O	Tag which specifies true, false or N/A to indicate whether the ARQC specified in the online message was validated, failed to validate or where no validation was attempted.
MACValidated	Tag	O	Tag which specifies true, false or N/A to indicate whether the MAC specified in the online message was validated, failed to validate or where no validation was attempted.
CVC3Track1Validated	Tag	O	Tag which specifies true, false or N/A to indicate whether the CVC3 specified in the track-one data present in the online message was validated, failed to validate or where no validation was attempted.
CVC3Track2Validated	Tag	O	Tag which specifies true, false or N/A to indicate whether the CVC3 specified in the track-two data present in the online message was validated, failed to validate or where no validation was attempted.
ToolComment	Tag	O	An application-generated comment related to the online message.

<b>Field</b>	<b>Block/Tag/Attribute</b>	<b>M/O/C</b>	<b>Description</b>
Date-Time	Tag	M	Date and time when the first byte of the online message first appeared over its connection. Expressed in UTC.
</MessageInfo>	Block-end	M	
<FieldList>	Block-start	M	Contains zero or more Field objects.
<Field>	Block-start	M	A single field within an online message or data block within an online message exchange.
ID	Attribute of tag < <i>Field</i> >	M	Tag which specifies the name of the field or data block, typically as it is referred to by the specification document governing the online message format. This value should be short and unique so that the value specified in the field may be referred to unambiguously, within the scope of the governing specification document, by any code which needs to locate and access the value. It is intended that the value of this attribute is only used internally by software to identify the field uniquely, and is not user-visible.
FriendlyName	Tag	M	Tag which specifies the user-readable name of the field. It is intended that the value of this field is user-visible only, and is not used internally by software to identify the field uniquely.
SearchSymbol	Tag	O	If present, SearchSymbol specifies the identifier and value of an EMVCoL3-defined ‘common’ field. The purpose of this tag is to permit easy searching within online message files for specific online messages. An OnlineMessage entry need not specify any common fields.
Name	Attribute of <SearchSymbol>	O	Tag which specifies the identifier of a common field. The common field identifiers defined by EMVCoL3 are PAN, AMOUNT, ARQC, ARPC, TERMINALID, MERCHANTID, ACQUIRERID, STAN, RRN, PROCESSINGCODE and UNPREDICTABLENUMBER.
Value	Attribute of <SearchSymbol>	O	Tag which specifies the value of a common field, in a format which can be presented to, and selected by, the user.
EMVData	Tag	O	Tag which specifies that the enclosing Field is an EMV data element. For each separate EMV data element, this tag should be present within the enclosing OnlineMessage block once only.
Tag	Attribute of <EMVData>	C	Attribute which specifies, in hexadecimal, the tag identifier of the EMV data element
Name	Attribute of <EMVData>	C	Attribute which specifies the name of the EMV data element. Where a tag is used by multiple card schemes, the relevant card scheme name should be included together with the Name value to allow the tag to be uniquely identified.

Field	Block/Tag/Attribute	M/O/C	Description
Format	Attribute of <EMVData>	C	<p>Attribute which specifies the manner in which the EMV data element is presented in the associated FieldBinary tag. The value specified must be one of the following three values, to indicate the associated content:</p> <ul style="list-style-type: none"> <li>o "V" – the value of the EMV data element;</li> <li>o "LV" – the data length identifier and the value of the EMV data element;</li> <li>o "TLV" - the tag identifier, the length and the value of the EMV data element;</li> </ul> <p>The tag identifier and the data length identifier, if either or both are specified, must be encoded according to the standard BER-TLV encoding rules.</p>
FieldType	Tag	M	Tag which specifies the type of the field or subfield for informational purposes only. For this file version, EMVCoL3 recommends the use of field types as specified in the same format as used in the ISO 8583-87 documentation – eg, “ANS..17”, “N5” etc.
PrefixBinary	Tag, human-readable binary data	C	Tag which specifies the raw hexadecimal data for a field or subfield prefix, if one is present (example – '00 1A', for a two-byte field length specifier, where the value is expressed in hexadecimal)
PrefixViewable	Tag	C	Tag which specifies the viewable form of the field or subfield prefix, if one is present, and typically expressed in decimal notation (example '26')
FieldBinary	Tag, human-readable binary data	M	Tag which specifies the raw, hexadecimal data for the field or subfield. An L3 test tool which accesses the fields within the online message shall use the data specified in this tag (example 'F0 F1 F0 F0')
FieldViewable	Tag	M	Tag which specifies the user-readable form of the field (example '0100'). Test code should not rely upon values specified in this tag.
ToolComment	Tag	M	An application-generated freeform text string which specifies a comment which typically describes the contents of the field in more detail (example 'Numeric amount of 100').
ToolCommentLevel	Tag	O	An application-generated tag which specifies whether the accompanying ToolComment tag is informational (Info), warning (Warn) or error (Error) condition for the comment. If this tag is not specified, a value of Info can be assumed.
<FieldList>	Block-start	M	Specifies zero or more subfields contained within a parent field. This structure is recursive to an arbitrary degree, so the online message format can support fields and subfields to an arbitrary depth. The format of the FieldList block is identical to the format of the parent FieldList block.
</FieldList>	Block-end	M	

<b>Field</b>	<b>Block/Tag/Attribute</b>	<b>M/O/C</b>	<b>Description</b>
</Field>	Block-end	M	
</FieldList>	Block-end	M	
</OnlineMessage>	Block-end	M	
</OnlineMessageList>	Block-end	M	
Signature	Tag	M	Specifies the XML signature as per the EMVCoL3-specified terminal-7816 log format.
</EMVCoL3OnlineMessageFormat>	Block-end	M	

**Note:** PrefixBinary and FieldBinary appear in human-readable binary format. This format specifies hexadecimal data in the usual [0-9A-Fa-f][0-9A-Fa-f] format, but can include spaces between byte values in order to make the data more easily comprehensible to human readers. Examples of valid human-readable binary data include “C0DECAFE”, “0102 0304 05 06”, “DE AD BE EF”, “DE ad C0 de”. Examples of invalid human-readable binary data include “D EA DB EE F” (spaces included at nibble boundary, not byte boundary) and “DE AD BE E” (final byte is incomplete).

**Table 4.15: Online message sample**

```
<EMVCoL3OnlineMessageFormat>
<LogDetails>
  <Date-Time>2015-06-01T03:14:15Z</Date-Time>
  <LoggingTool>
    <ProductName>ACME Test Tool</ProductName>
    <ProductVersion>1.0</ProductVersion>
  </LoggingTool>
  <SchemaSelectionIndex>1.0</SchemaSelectionIndex>
  <Reference>Comment concerning the online messages logged here</Reference>
</LogDetails>
<ConnectionList>
  <Connection ID="SCHEME-SIMULATOR">
    <Protocol>
      <FriendlyName>Visa BASE I</FriendlyName>
      <SymbolicName>VISABASEI</SymbolicName>
      <VersionInfo>2015.01</VersionInfo>
    </Protocol>
    <TCPIPParameters>
      <Address>192.168.100.1</Address>
      <Port>1234</Port>
      <Header>4 bytes, PDP-order</Header>
      <Client>false</Client>
      <Format>ASCII</Format>
    </TCPIPParameters>
  </Connection>
  <Connection ID="CUSTOMER-SYSTEM">
    <Protocol>
      <FriendlyName>Visa BASE I</FriendlyName>
      <SymbolicName>VISABASEI</SymbolicName>
      <VersionInfo>2015.01</VersionInfo>
    </Protocol>
    <TCPIPParameters>
      <Address>192.168.100.2</Address>
      <Port>4321</Port>
      <Header>4 bytes, PDP-order</Header>
      <Client>true</Client>
      <Format>ASCII</Format>
    </TCPIPParameters>
  </Connection>
</ConnectionList>
<OnlineMessageList>
  <ToolComment>Client to server link is now active</ToolComment>
  <ToolComment>Client is sending a simple authorization request</ToolComment>
  <OnlineMessage Class="REQUEST" Source="CUSTOMER-SYSTEM" Destination="SCHEME-SIMULATOR">
    <RawData>08 49 00 00 00 00 00 01 00 00 03 01 23 00 09 9F 27 01 80 9F 36 02 00 01</RawData>
    <MessageGroup>BA0134DD-6194-4F04-A108-9065F8DA355A</MessageGroup>
    <MessageInfo>
      <PINValidated>N/A</PINValidated>
      <PINKey>12 34 56 78 90 AB CD EF</PINKey>
      <ToolComment>This is a comment concerning the request</ToolComment>
      <Date-Time>2015-06-10T03:14:15.926Z</Date-Time>
    </MessageInfo>
    <FieldList>
      <Field ID="PAN">
        <SearchSymbol Name="PAN" Value="49000000000000000000000000000000"/>
        <FriendlyName>Primary Account Number</FriendlyName>
        <FieldType>N..16</FieldType>
        <PrefixBinary>08</PrefixBinary>
        <PrefixViewable>08</PrefixViewable>
        <FieldBinary>49 00 00 00 00 00 00 00</FieldBinary>
        <FieldViewable>49000000000000000000000000000000</FieldViewable>
        <ToolComment>BIN belongs to Bank A</ToolComment>
      <FieldList/>
    </Field>
    <Field ID="DE003">
      <FriendlyName>Processing Code</FriendlyName>
      <FieldType>N3</FieldType>
    </Field>
  </OnlineMessage>
</OnlineMessageList>
```

```
<FieldBinary>01 00 00</FieldBinary>
<FieldViewable>010000</FieldViewable>
<FieldList>
  <Field ID="DE003S01">
    <FriendlyName>Transaction Type</FriendlyName>
    <FieldType>N1</FieldType>
    <FieldBinary>01</FieldBinary>
    <FieldViewable>01</FieldViewable>
    <ToolComment>[01] Cash Disbursement</ToolComment>
    <FieldList/>
  </Field>
  <Field ID="DE003S02">
    <FriendlyName>Account Type (from)</FriendlyName>
    <FieldType>N1</FieldType>
    <FieldBinary>00</FieldBinary>
    <FieldViewable>00</FieldViewable>
    <ToolComment>[00] Not applicable or not specified</ToolComment>
    <ToolCommentLevel>WARN</ToolCommentLevel>
    <FieldList/>
  </Field>
  <Field ID="DE003S02">
    <FriendlyName>Account Type (to)</FriendlyName>
    <FieldType>N1</FieldType>
    <FieldBinary>00</FieldBinary>
    <FieldViewable>00</FieldViewable>
    <ToolComment>[00] Not applicable or not specified</ToolComment>
    <ToolCommentLevel>WARN</ToolCommentLevel>
    <FieldList/>
  </Field>
  <FieldList>
  </Field>
</Field>
<Field ID="DE004">
  <SearchSymbol Name="AMOUNT" Value="12300"/>
  <FriendlyName>Transaction Amount</FriendlyName>
  <FieldType>N3</FieldType>
  <FieldBinary>01 23 00</FieldBinary>
  <FieldViewable>012300</FieldViewable>
  <ToolCommentLevel>INFO</ToolCommentLevel>
  <FieldList/>
</Field>
<Field ID="DE023">
  <FriendlyName>PAN Sequence Number</FriendlyName>
  <FieldType>N2</FieldType>
  <EMVData Tag="5F34" Name="PSN" Format="V"/>
  <FieldBinary>09</FieldBinary>
  <FieldList/>
</Field>
<Field ID="DE055">
  <FriendlyName>EMV Data</FriendlyName>
  <FieldType>B..255</FieldType>
  <FieldBinary>9F 27 01 80 9F 36 02 00 01</FieldBinary>
  <FieldList>
    <Field ID="DE055.01">
      <FriendlyName>Cryptogram Information Data</FriendlyName>
      <FieldType>B4</FieldType>
      <EMVData Tag="9F27" Name="Cryptogram Information Data" Format="TLV"/>
      <FieldBinary>9F 27 01 80</FieldBinary>
      <FieldList/>
    </Field>
    <Field ID="DE055.02">
      <FriendlyName>EMV Data</FriendlyName>
      <FieldType>B5</FieldType>
      <EMVData Tag="9F36" Name="Application Transaction Counter" Format="TLV"/>
      <FieldBinary>9F 36 02 00 01</FieldBinary>
      <FieldList/>
    </Field>
  <FieldList>
  </Field>
</FieldList>
</OnlineMessage>
<ToolComment>Client is sending a simple authorization request</ToolComment>
</OnlineMessageList>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
```

```
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <DigestValue>tp4w4QGL9e0aFBORXAM4l4OQWnl=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>WBD1i1REbPmaqykvB5ealvZhBit902cHKxl4SY0gfMcGMIxV1Yq3mwxNG1ct1Bt8niKc7VNsyit6/pbtJOkIrsBU
cKuDiWgHrSWmJCBDDQQ8tZwafKoLYbZK5AV3WLquqHrX+iO6KeeFkTHVmRUIXGbO1vXAAyz3bWHvG5in9O4=</Signature
eValue>
<KeyInfo>
  <KeyName>ACME TEST TOOLS LTD Key 01</KeyName>
</KeyInfo>
</Signature>
</EMVCoL3OnlineMessageFormat>
```



## 4.8 TSE Validation Report Files

The TSE Validation Report file is using a formal XML syntax. It allows providing comments and status updates on the test results.

The TSE Validation Report file is provided as a **.tser file** (a renamed .xml file). The structure of the file is similar to the embedded xml files in the .tse file.

### 4.8.1 TSE Validation Report structure

Section - Block tag - Data field tag	Description	L3TSE Requirement
<ValidationInfo>	L3TSE Validation Report file identifier	
...<Context>	Single entry	
....<TrackingNo>	This is the tracking number of the .tse file that has been reviewed	
....<OriginalFilename>	This is the name of the .tse file that has been reviewed	
....<RuleSetName>	These fields record the version of CSV rules present in the .tse file at the time of review.	If the .tse file has been updated to a newer version whilst the review was taking place then L3TSE will warn the user so that they can decide not to proceed with the automatic processing.
....<RuleSetSeries>		
....<RuleSetBuild>		
....<ReviewDate>	This field records the date and time of the validation.	
....<ReviewStatus>	This is set to either "pass" or "fail" depending on whether or not the validation was successful.	
....<ReviewText>	This field may contain an html validation report describing the issues found during testing.	L3TSE will display this information when processing the validation report.
...<Observations> ...<Test> ....<TestName> ....<Observation>	When processing validation reports, L3TSE has the ability to update the <ReviewComments> information for the test and to make the comments visible on the user interface when the test is opened.	
...<Answers> ...<Response> ....<Name> ....<Answer>	The validation report format allows for new answers to be provided for one or more questions.	If new answers are provided then L3TSE will update the necessary question responses and switch to question entry mode. New answers are only

Section - Block tag - Data field tag	Description	L3TSE Requirement
...<Results> ....<Test> .....<TestName> ....<MajorVersion> ....<CheckNo> ....<StepNo> ....<Result>	This section allows a reviewer to override the test results of individual test steps. Typically all steps of a test that has to be repeated will be cleared or set to "fail" however the format does allow for individual steps to be updated.	relevant when the <ReviewStatus> is "fail".  When processing failed validations, L3SE will provide the option for the user to update the status of test steps to match the entries in the validation report.

## Annex A Common L3 Terminologies

Below is a working glossary of commonly-used terms and acronyms with the L3 testing process.

A	
A	Alpha
AAC	Application Authentication Cryptogram
Acquirer	Financial institution which passes on transaction data from merchant to payment system.
ADA	Application Default Action
ADF	Application Definition File
ADM	Automated Dispensing Machine. An unattended device that accepts payment for dispensed goods, such as fuel, has electronic capability, and accepts PINs, but does not disburse currency.
ADVT	Acquirer Device Validation Toolkit. The Visa Inc and Visa Europe terminal integration testing process.
AEF	Application Elementary File
AEIPS	American Express ICC Payment Specification. The Amex implementation of EMV.
AFD	Automated Fuel Dispenser
AFL	Application File Locator
AID	Application Identifier
AIP	Application Interchange Profile
an	alphanumeric
ans	alphanumeric special
Application Protocol Data Unit (APDU)	The communication format used between the chip card and the payment application on a card acceptance device. This format is defined in ISO specification 7816-4.
ARPC	Application Response Cryptogram
ARQC	Application Request Cryptogram
ATC	Application Transaction Counter
ATM	Automated Teller Machine. An unattended device that has electronic capability to send transactions online for authorization, accepts PINs, and disburses currency.
AUC	Application Usage Control
Automated Dispensing Machine	An unattended device that accepts payment for dispensed goods, such as fuel, has electronic capability, and accepts PINs, but does not disburse currency.

Automated Teller Machine (ATM)	An unattended device that has electronic capability to send transactions online for authorization, accepts PINs, and disburses currency.
<b>B</b>	
b	Binary
<b>C</b>	
CAM	Card Authentication Method
Card Acceptor Terminal	See “ <i>Terminal</i> ”
Card/Terminal Log	A capture of the data exchanged between the card/card simulator and the acceptance device. Typically provided in Application Protocol Data Unit (APDU) format (presently ISO 7816-3 and ISO 14443).
Card/Terminal Log Validation	A description of the requirements for validating the elements and content of the Card/Terminal Log during terminal integration testing. It is defined by a Test Session files.
Cardholder Activated Device	An unattended device, such as an automated dispensing machine, self-service device, or limited amount device that is not an ATM. See also: <b>UAT</b> , Cardholder Activated Terminal.
CAT	Customer Activated Terminal
CDA	Combined Dynamic Data Authentication
CDOL	Card Risk Management Data Object List
Certification	Validation that the format, function and content of authorization messages executed from a defined test plan adheres to a provided specification and set of business rules.
CID	Cryptogram Information Data
cn	Compressed numeric (also known as BCD)
Comma-separated Values (CSV)	A format that stores tabular data (numbers and text) in plain-text form (i.e. a sequence of characters, with no data that has to be interpreted instead, as binary numbers). A CSV file consists of any number of records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab. Usually, all records have an identical sequence of fields.
Contact transaction	An interaction between a chip application and a device using the physical electrical interface, as defined in [EMV Book 1].
Contactless IC Terminal Check for Implementation (TCI-CL)	The JCB contactless IC terminal integration testing process.
Contactless transaction	An interaction between a chip application and a device using the radio frequency wireless interface, as defined in [EMV CL].

Customer Activated Terminal (CAT)	See ‘Cardholder Activated Device’
CVM	Cardholder Verification Method
CVR	Cardholder Verification Result
<b>D</b>	
DDA	Dynamic Data Authentication
DDF	Directory Definition File
DDOL	Dynamic Data Authentication Data Object List
DES	Data Encryption Standard
DGI	Data Group Identifier (used by the Card Personalizer only)
D-PAS Acquirer Terminal E2E	Terminal integration testing process for Discover.
DKI	Derivation Key Index
D-PAS	Discover Payment Application Specification
<b>E</b>	
EMV	A term referring to certain technical specifications developed and maintained by EMVCo and/or technologies conforming to such specifications.
EMVCo LLC (EMVCo)	The limited liability company that manages, maintains, and enhances the EMV Specifications. Current Members of EMVCo are American Express, Discover, JCB, Mastercard, UnionPay and Visa.
EMV Specifications	Technical specifications developed and maintained by EMVCo to facilitate worldwide interoperability and acceptance of secure payment transactions, including the requirements described in the bulletins available on the EMVCo website.
<b>F</b>	
FCI	File Control Indicator
<b>G</b>	
GPO	Get Processing Options
<b>H</b>	
Hex.	Hexadecimal
Host Authorization Message	The transaction message initiated from the device and sent online via the acquirer and network to the issuer, processor or Payment system for transaction authorization and back to the device. This is a series of message formats, currently implemented by more than one protocol.
<b>I</b>	
IAC	Issuer Action Code
Interoperability	The ability of all card acceptance devices to accept and read all chip cards that are properly coded and personalized.

<b>ISO</b>	International Organization for Standardization
<b>Issuer</b>	A financial institution which makes a card or other payment method available to a consumer or business and which controls the account to which the card or payment method is linked.
<b>K</b>	
<b>Kernel</b>	A piece of software implementing the set of functions required to support the EMV specifications. The kernel may contain device drivers, interface routines and security and control functions. The kernel has to be sufficiently separable from the other software elements constituting the complete terminal application that it can have its own digital signature and be tested separately from any specific deployed version of the terminal implementation.
<b>L</b>	
<b>Limited Amount Device</b>	An unattended device that has data capture-only capability, and accepts payment for items such as parking garage fees, road tolls, etc.
<b>L3 CTT</b>	L3 Consolidated Test Tool – a L3 tool, qualified by EMVCo, for inclusion of the following L3 tool components; L3 Test Selection Engine (L3 TSE), L3 Test Tool (L3 TT), L3 Card Simulator (L3 CS)
<b>L3 LoQ</b>	L3 Letter of Qualification – a formal letter issued to a L3 Test Tool provider by EMVCo, on successful completion of the L3 qualification process.
<b>L3 QSP</b>	L3 Test Tool qualification service provider – entity accredited by EMVCo to perform the qualification services on L3 test tools submitted by third-party vendors for qualification.
<b>L3TG</b>	Level 3 Testing Group. The new EMVCo-approved name assigned to the Terminal Integration Task Force (TITF).
<b>M</b>	
<b>Mobile Point-of-Sale/Mobile Point-of-Service (MPOS or mPOS) device</b>	A smartphone, tablet or dedicated wireless device that performs the functions of a cash register or electronic point of sale terminal (POS terminal), usually in conjunction with a card reader and PIN pad device which can encrypt transaction data
<b>M-TIP</b>	Mastercard Terminal Integration Process. The Mastercard terminal integration testing process.
<b>O</b>	
<b>Offline-capable</b>	A transaction acceptance device that has the ability to process transactions offline, making use of the EMV offline authorization functionality.
<b>Offline-only</b>	A transaction acceptance device that is only able to process transactions offline. For example, low value POS or road toll terminals.
<b>Online-capable</b>	A transaction acceptance device that is able to send transactions to the issuer or processor for authorizations.

Online-only	A transaction acceptance device that requires that all transactions be sent online for authorization. For example, ATM.
<b>P</b>	
PA-DSS	Payment Application Data Security Standard. The PCI SSC standard relating to secure storage of transaction data.
PAN	Primary Account Number
PAN Key Entry	A manual procedure in which the merchant uses a device key pad to enter the PAN embossed on a card in order to process a transaction.
Pass Criteria	A Test Plan-defined description of an expected result for a successful outcome or conclusion of a test case.
Pass Criteria File	The file (in CSV format) that embeds the Payment system-defined pass criteria for each test case.
PCI SSC	Payment Card Industry Security Standards Council.
PDOL	Processing Options Data Object List
Personalization	For Chip cards, the process of applying both cardholder and Payment system-specific data to the card in preparation for its use.
PIN	Personal Identification Number
PIX	Proprietary Application Identifier Extension
PK	Public Key
PKI	Public Key Infrastructure
Point of Sale/Point of Service (POS)	The physical location where a merchant or acquirer in a face-to-face environment or an unattended device completes a transaction.
Point-of-sale Device	A card accepting device used to process sale transactions.
Point-of-service Device	A card accepting device used to process transactions. Can be any type of transaction.
PSE	Payment Systems Environment
<b>R</b>	
RFU	Reserved For Future Use
RID	Registered Application Provider Identifier
<b>S</b>	
SDA	Static Data Authentication
<b>T</b>	
TAC	Terminal Action Code
TAD	Transaction Acceptance Device. Another word for card acceptance device or terminal. Applies to POS devices and ATMs.
TC	Transaction Certificate
TDOL	Transaction Certificate Data Object List

Terminal	The device used in conjunction with the Chip card at the point of transaction to perform a financial transaction. The terminal incorporates the interface device and may also include other components and interfaces such as host communications.
Terminal Check for Implementation (TCI)	The JCB terminal integration testing process.
Terminal configuration	A description of the features and parameters on the acceptance device under test. For example, it might include the EMV-defined terminal types, supported interfaces , etc.
Terminal integration testing	A process implemented and managed by the respective Payment systems, aimed at providing a level of assurance that Payment system-specific requirements and recommendations are being implemented in contact or contactless chip acceptance device designed specifically to interoperate with the Payment Systems' networks.
Terminal Integration Task Force (TITF)	Task Force established by EMVCo in 2013 with the purpose of examining each of the Payment system's terminal integration processes, in order to determine the possibilities of aligning key elements of these processes.
Test Card Image	An electronic representation of the data on a physical card
Test Case	A single test scenario of a Test Plan, corresponding to one action by the tester. It is defined by a Payment System for execution of terminal integration or Host message testing.
Test Case Name	The name associated with a specific test case.
Test Case Number	A unique test case identifier.
Test Case Objective	A description of the reasons for a given test case. This is based on Payment System rules.
Test Plan	A Payment system-developed and managed set of test criteria that defines the requirement for Terminal Integration or Host Message testing. This may either take the form of a text document, or a machine-readable file.
Test Procedure	A description of the steps required in order to execute a specific test case.
Test Tool Qualification	The process undertaken by each Payment system to provide themselves, tool vendors and clients with a level of assurance that the tools being used by clients to execute terminal integration testing, will do so in compliance with Payment system requirements.
Third Party Processor	A party which is appointed by a merchant or financial institution to provide payment transaction processing services. In this capacity they are <i>not</i> acting as a merchant, acquirer, issuer or payment system.
TLV	Tag-Length-Value
Transaction Completion	An EMV definition for the successful closing of transaction processing. The completion function is always the last function in transaction processing, and must occur unless the transaction is terminated prematurely by error processing.

TSI	Transaction Status Information
TVR	Terminal Verification Result
<b>U</b>	
Unattended Acceptance Device (UAT)	A cardholder-operated device that reads, captures, and transmits card information in an unattended environment. Also known as ‘Customer Activated Terminal’ (CAT) or “Cardholder Activated Device”
UnionPay IC Card Test Guide for Acquirers	The terminal integration testing process for UnionPay.
User Validation	A Test Plan-defined description of the requirements for the user/tester to validate responses from the device under test.
<b>V</b>	
var.	Variable
VAR	Value-added Reseller
<b>X</b>	
Extensible Mark-up Language (XML)	A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
XSD	XML Schema Definition

\*\*\* END OF DOCUMENT \*\*\*