EMV®
Level 3 (L3) Testing Framework

Implementation Guidelines

Version 1.2
March 2023

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	Date	Description
V1.0	March 2017	First public release of document.
V1.1	April 2019	• Include explicit requirements for TSE Test Tool Vendors in **Section 3.2: L3 TSE Requirements**. • **Section 4.1.1: Test Set File Considerations** indicates that "the headers can appear in any order". • **Section 4.1.3: Question Definition File Format –** introduction of an "Attributes" field. • **Section 4.2.1: TestRunInfo.xml** – Inserted <PSI> as part of the TrackingNo's unique identifier field. • **Table 4.23: Online Message log Format** – updated the Description of the FieldBinary field. • Other editorial updates throughout.
V1.2	March 2023	**Clarifications and corrections not related to EMV Book C-8:** • **Section 2: L3 Test Environment Architecture** - added reference to the TSER-package. • **Figure 2-1: Test Environment Architecture –** replaced Payment System by Participant Systems. • **Section 3.1: Definitions** – added a note after *value*, updates to some of the examples, Applicability condition and added the definition of a Project. • **Section 3.2: L3 TSE Requirements** – updated or added Req 2.4, 5.0, 6.1.1, 12.0, 12.5, 15.2, 14.4, 15.0, 15.5, 17.0, 17.1, 18.4, 18.5, 18.6, 18.7 and 18.8. • **Section 3.3: L3 TT Requirements** – include explicit requirements for TT Test Tool Vendors. • **Section 3.4: L3 CS Requirements** – include explicit requirements for CS Test Tool Vendors. • **Section 4: EMVCo L3 Detailed Formats** – added a general description, requirements and clarifications. • **Section 4.1.1: Test Set File Considerations** - updated the following sections: introduction, Bit Definition fields, Applicability of Context-related Data, Processing the TSE Test Set Files, TSE Test Set Files Format, HTML codes in Text field and Naming conventions, and moved the Character Substitutions section to section 4.10. • **Section 4.1.2: Test Selection File Format** – updates to the Context Expression 1.. n Data field. • **Section 4.1.3: Question Definition File Format** – updates to the Common Questions section, Type Data field and validation=email Mode.

		• **Section 4.1.4 Error Definition File Format –** updates to ErrorType Data field. • **Section 4.1.5: Suggestion Definition File Format** – added and updated some rules, and updated SuggestionType Data field in Table 4.9: Suggestion Definition File Format. • **Section 4.1.8: Test Case File Format** – updates to Cards Data field. • **Section 4.1.11: Manifest File Format** – added clarification to File format, updated the example (included the Version) and added the description of each parameter. • **Section 4.2: TSE Test Session Files** – updated the Manifest file to be mandatory, added clarification on the xml prolog, added requirements and a table to indicate which component a L3 TT can write or read. • **Section 4.2.1: TestRunInfo.xml** - reformatted the table, added L3TSEFIGVersion and L3TTFIGVersion fields, and clarified the TrackingNo, MachineID, RuleSetName, RuleSetSeries, RuleSetBuild, RuleSetMode and Mode fields. • **Section 4.2.2: RuleSet.xml** - reformatted the table, added clarification in introduction, added missing tags, added a description and requirement column for the fields including clarifications, and moved the Substitutions section to section 4.10. • **Section 4.2.3: TestRun.xml** - reformatted the tables and added a description and requirement column for the fields when created or updated including clarifications. • **Section 4.2.4: Selected.xml** - reformatted the table and added clarifications to the description and requirement column. • **Section 4.3: Tool Pass/Fail Automation Criteria** – updated the following lines: isMandatory, DataItem, DataFormat, Operator, Value, ActionIfTrue and ActionIfFalse. • **Section 4.4: Test Report** – added an example figure and added prefixes for Online Message log and Card to Terminal log filenames. • **Section 4.5: Test Card Image File Format** – updated the structure of this section and included clarifications. • **Section 4.5.2: Test Card Image Format** – added tags L3FIG version and L3PF version, and format specification table including clarifications in the Description and Requirement column. • **Section 4.6: Card Terminal log** Format **–** updated the structure of this section and included clarifications. • **Section 4.6.2: Card Terminal log Format Requirements –** added a format specification table including clarifications in the description and requirement column, and the following four new tags: L3FIGVersion, L3PFVersion, CardId and CardVersion. • **Section 4.6.4: Signature -** Signature support is optional.

		• **Section 4.7.2: Online Message log Format** – added L3OMLVersion field and missing closing DialupParameters, and made clarifications to the following fields: DialupParameters, SymbolicName, Class, RawData, PINValue, ID, FieldBinary, FieldViewable,ToolCommentLevel, ToolComment and Signature. • **Section 4.8: TSE Validation Report Files** – clarifications to the Introduction and the TSER file xml Format Specification sections. • **Section 4.9 Cryptographic Processing -** new section including clarifications related to hash calculation and example. • **Section 4.10: Character Substitution** – new section including clarifications related to the character substitutions. • **Annex A:** added and updated some terms. • **Annex B:** new annex to define tool pass/fail automation criteria for data item. • Other minor editorial updates throughout the document. **New Requirements related to EMV Book C-8:** • **Section 3.4: L3 CS Requirements** – refer to new requirements in section 7.0. • **Section 4.5.2: Test Card Image Format** – new attribute kernelID in <TerminalRequest> tag and new tag <ECCKeys> in <Crypto> tag. • **Section 4.6.2: Card Terminal log Format Requirements** – new attribute in <Response> tag.

Contents

Figures

Tables

1 Background

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

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 Participant 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, Participant 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 Online 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 Participant Systems) with a convenient means of preparing applicable Test Session files for individual Participant 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 Participant Systems. There may be one or more TSEC-packages per Participant 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 Participant 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 Participant 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.
- Import the machine-readable Validation Report provided as a TSER-package. It is used to send the feedback (i.e., pass/fail pass criteria status) of the session validation in a file that can be loaded in the TSE.

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 Participant 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

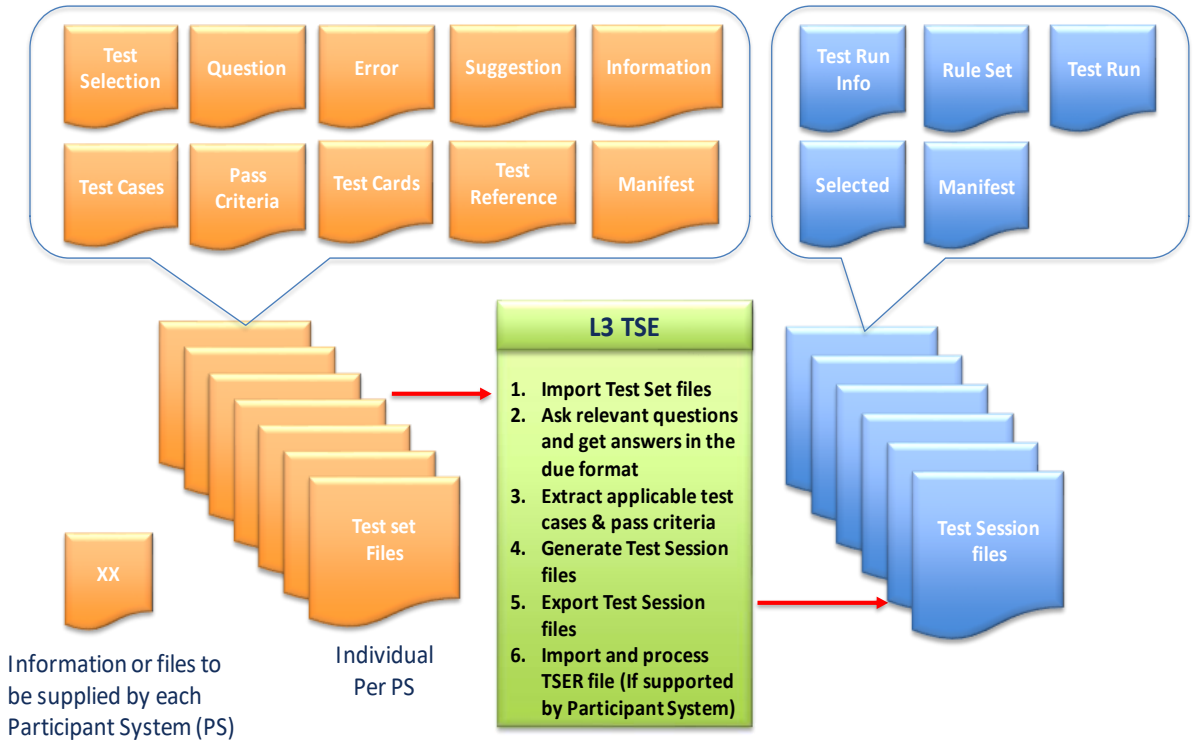
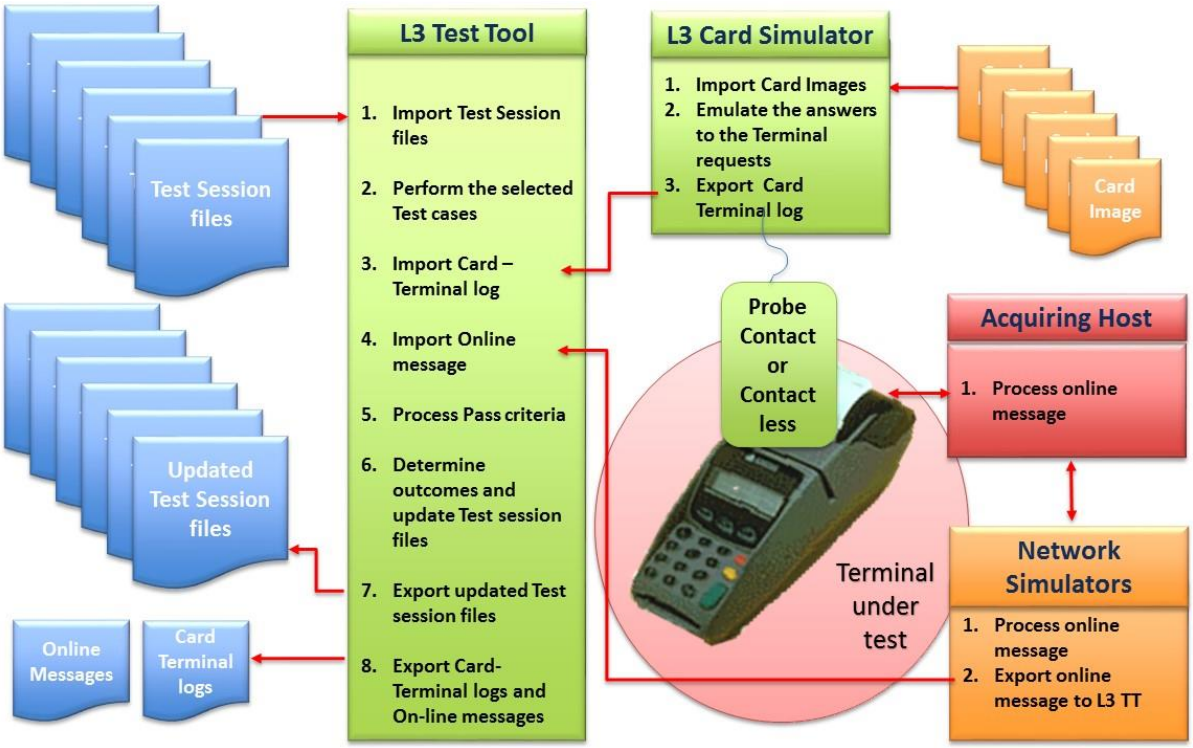


Figure 2-2: L3 TT and L3 CS Environment Architecture



3 L3 Component Requirements

3.1 Definitions

Terminal Integration Context

Describes the context required to create a test plan for all applicable Participant 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 Participant System might use their own criteria names. The Participant 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 (format: YYYYMMDD).

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.

Note: <value> in a context '<criteria name> <operator> <value>' cannot be empty.

Examples:

- CL_CVM_Limit = 0
- CL_Transaction_Limit = CL_CVM_Limit (i.e., both CL_Transaction_Limit and CL_CVM_Limit are Questions defined as varNumber)

- Acquirer_Country == Deployment_Country (i.e., both Acquirer_Country and Deployment_Country are Questions defined as varList or varSet)

  Note: for a varSet, if Acquirer_Country = [France] [Italy] and Deployment_Country = [Italy] [France], then Acquirer_Country == Deployment_Country is TRUE. If Acquirer_Country = [France] and Deployment_Country = [Italy] [France], then Acquirer_Country == Deployment_Country is FALSE.
- 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 errors: the error shall be considered as not applicable.

* Otherwise: it shall be ignored from the logical expression.[1]

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 Format.

Timestamp

Always in ISO-8601 format unless specified differently.

Project

A project is defined as the testing of one particular Terminal Integration solution deployed across one or several Participant System(s).

[1] 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 L3 TSE Requirements

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

Table 3.1: L3 TSE Requirements

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	• L3TSE shall be able to import the csv files with headers presented in any order and ignore any unknown csv header.	M
2.0	L3TSE shall only allow import of a TSEC that has a correct manifest file and shall check that:	M
2.1	• There are no files in the Test Set files that are not in the manifest file,	M
2.2	• All files listed in the manifest file shall be present,	M
2.3	• For each file the hash shall match,	M
2.4	• The manifest file should have the correct signature (if present).	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 section 4.10.1.	M
5.0	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). It is not acceptable when the HTML codes (like <b>) are shown to the user. When the L3TSE is processing the HTML codes defined in the Information Report (info.csv), the L3TSE shall make use of CSS style sheets in the TSEC-package, if present. **Note:** example of syntax used in the manifest file: Doc_Style=00_EMVCo_screenstyle.css:checksum.	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

Req	Requirement	M/O/C
6.1	• L3TSE shall evaluate the Question applicability for each of the questions to understand whether the question shall be presented to the user.	M
6.1.1	- The answer to a previously presented question with attribute "Common" shall be treated as a suggestion within the same project. It shall be handled as a "suggest" SuggestionType. Suggestions originating from common answers shall be treated after regular suggestions.	M
6.2	• L3TSE shall check the Suggestion file, to understand whether the questions in the group come with applicable suggestions. - Suggestions shall be shown to the user. - It shall be possible for the user to override suggestions. - 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. - Answers that are highlighted to be removed shall no longer be visible to the user. - In case multiple rules apply follow the guidance in section 4.1.5. - 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).	M
6.3	• The group of questions shall be presented in the following way: - L3TSE shall present the questions in the order they appear in the Question definition file.	M
6.4	• Questions that are grouped shall be presented together under the group prompt as specified in the Question definition file.	M
6.5	• L3TSE shall present help text available at group level and at question level to the user.	M
6.6	• 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.	O

Req	Requirement	M/O/C
6.7	• The ways in which the user can answer to questions shall respect the different answer types indicated in the Question definition: - A varBoolean shall only allow a Boolean choice. - A varNumber shall only allow a Numerical value. - A varList shall only allow one answer to be selected. - A varSet shall allow multiple answers to be selected. - An optional question (indicated by the mode field) shall not force an error if left unanswered.	M
6.8	• L3TSE can use the guidance given in the mode field indicated in the Question definition. - A drop_down should be represented by a drop box. - 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. - If the mode explicitly mentions the number of lines / element (:N) this should be respected.	M
6.9	• After having received the answers to the questions in the group L3TSE shall: - Check whether the answer to the question has the correct type (e.g., Boolean, number, string, email address[2]).	M
6.10	- Update the Terminal Integration Context.	O
6.11	- Process the full Error file and display errors or warning messages as defined. - 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). - 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.	M

[2] Indicated by the mode in the question definition file, see section 4.1.3.

Req	Requirement	M/O/C
7.0	Finally, L3TSE shall **present the Information Report** 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	• It shall present the headers at the right level, indicated by the HeadingLevel, followed by the indicated content from the Information Report file.	M
7.2	• All placeholders (see section 4.1.6) should be filled with the correct values.	M
7.3	• 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.	M
8.0	L3TSE shall allow the **Question Review Process**:	M
8.1	• 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.	M
8.2	• 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. **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.	M
8.3	• When all questions are answered, L3TSE shall *present the Information Report* for the updated Terminal Integration Context as defined above.	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 Participant 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 export and import his current context when creating or modifying (see below) the (preliminary) TSE Test Session files:	M

Req	Requirement	M/O/C
12.1	• If the MODE_Regression indicator is set, the Mode in the TestRunInfo.xml shall be put to MODE_Regression.	M
12.2	• Otherwise, if not all questions have been answered, the Mode in the TestRunInfo.xml shall be put to MODE_Question.	M
12.3	• Otherwise, the Mode in the TestRunInfo.xml shall be put to MODE_Info.	M
12.4	• The ChangeFlagDate in the same file shall reflect the current timestamp.	O
12.5	• L3TSE component shall update the TestRunInfo.xml with the version of the FIG, used by the test tool at the time of the test run.	M
13.0	L3TSE shall allow the user to **start a TSE Test Session review** in the following way:	M
13.1	• L3TSE allows a user to import a populated TSE Test Session file.	O
13.2	• 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.	M
13.3	• 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 *Question Review Process* as defined in Req 8.0: - If the Mode tag in TestRunInfo,xml is MODE_Regression then L3TSE shall set the MODE_Regression indicator. - L3TSE shall present the *Information Report* for the Terminal Integration Context as defined in Req 7.0.	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 Req 15.0)	M
14.1	• *Start a TSE Test Session review* as described in Req 13.0.	M

Req	Requirement	M/O/C
14.2	• 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.	M
14.3	• The user shall be shown a warning that s/he is only to allow to keep previous test results if s/he is sure the outcome of the tests are not depending upon anything that was changed.	M
14.4	• L3TSE updates the TSE Test Session files: - Update ChangeFlagDate in TestRunInfo.xml. - If the user decides to overrule previous test results, then completely refresh TestRun.xml as described in 4.2.3. - If the user decides to keep previous test results, then update TestRun.xml. i. Remove all Test Cases that are "Not executed" and no longer applicable. ii. Add Test Cases that are applicable and not yet in the TestRun.xml. 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. - Completely refresh Selected.xml as described in 4.2.4.	M
15.0	L3TSE shall allow the user to upgrade their TSE Test Session file to a higher build level using the PSI (P) and series (S) as a unique identifier. 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 Req **14.0**)	M
15.1	• L3TSE shall allow a user to upload a populated TSE Test Session file.	M
15.2	• L3TSE shall ask the user whether or not s/he wants to upgrade to the latest Build. If not, then the procedure as described in Req 14.0 applies, if yes, then continue as described below.	M
15.3	• The user shall be presented with all applicable questions not yet answered, or whose answers are no longer valid, like described in the *Question Review Process*.	M

Req	Requirement	M/O/C
15.4	• L3TSE then allows the user to export a new TSE Test Session file as described in section 4.2.[3]	M
15.5	• In the TestRunInfo.xml the file shall keep the original TrackingNo.	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	- *Start a TSE Test Session review* as described in Req **13.0**.	M
16.2	- L3TSE then allows the user to export a new TSE Test Session file as described in section 4.2, with one exception: - In the TestRunInfo.xml the file shall get a new TrackingNo, and the original TrackingNo should be put in the CopyOf field.	M
17.0	The L3TSE shall be able to display, at minimum, the content of the selected test cases:	M
17.1	• The selected test cases, on which additionally L3TSE may allow to apply filters indicated by the Attributes. • The related cards, including the tags defined in the Card File. • The Test Progress as indicated in the TestRun.xml. • The Name, Objective, Configuration, Applicable, Notes, Actions and Requirements. • For a Pass Criteria, the Commentary shall be displayed.	M

17.2	• Manual modification of the Test Progress by updating TestRun.xml (for customer that don't want to use an L3TT):
	  - Mark pass criteria pass/fail.
	  - Mark test cases pass/fail (with automation to do so based upon the set of pass criteria).
	  - Add logs, attachments, observations.
		M

[3] 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.

Req	Requirement	M/O/C
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	• L3TSE shall allow a user to upload a validation report (.tser).	M
18.2	• 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.	M
18.3	• L3TSE shall display the information present in the *ReviewText (formatted in html)*.	M
18.4	• L3TSE shall update the original .tse file with the info provided in the validation report (.tser), if confirmed by the user.	M
18.5	• When the XML Validation Report is loaded in the L3TSE, a Validation Report window shall be displayed with the below information: - The Tracking Number, the date <ReviewDate> of the report and the validation status: "pass" or "fail". - A free comment section <ReviewText>. - Test Review: an optional section listing the test cases containing Review Comments. - Question Review: an optional section listing the questions that may need to be reviewed.	M
18.6	• The user can decide to apply the changes to the test session. If a change is applied the TSE test session files shall be updated.	M
18.7	• When the Validation Report is applied, the L3TSE shall apply all the updates to the L3TSE session files and mark the updated test cases as changed.	M
18.8	• The Validation Report .tser file shall be added to the L3TSE session as a global attachment.	M

3.3 L3 TT Requirements

Table 3.2: L3 TT Requirements

Req	Requirement	M/O/C
1.0	L3TT shall be able to import L3 TSE file compliant with the file definition described in **section 4.2.**	M

Req	Requirement	M/O/C
1.1	To support backwards compatibility, the test tool shall be able to import a .tse file which, for example, may include additional data.	M
1.2	Hash values: L3TT shall check the TSE file Hash values and shall warn the user if one (or more) hash value(s) is (are) not correct. In that case the TSE file shall be further processed.	M
1.3	TSE file mode: The L3TT shall not import TSE files in the following modes and shall warn the user: • MODE_Question • MODE_Info • MODE_Regression The L3TT shall only import the TSE file in MODE_TestResult.	M
2.0	L3TT shall import and process L3 Card to Terminal logs following the pass/fail criteria described in **section 4.3** and compliant with the file definition described **section 4.6.**	M
2.1	Signature: L3TT shall not check the signature	M
3.0	L3TT shall import and process L3 Online Message logs compliant with the file definition described in **section 4.7**.	M
3.1	L3TT shall not check the signature.	M
3.2	The L3TT check validation shall only rely on Field ID and FieldViewable Tags and shall not block Log processing if any other XML Tag is absent or incorrectly formatted. If Field ID or FieldViewable Tags are absent or incorrectly formatted for a check that is in scope, the L3TT shall fail the validation of that particular check.	M
3.3	A L3 test tool which accesses the fields within the online message shall use the data specified in this tag to compare it with the Value of a Test Case Pass/Fail Criteria (refer to **section 4.3** for details) without applying any conversion.	M
4.0	L3TT shall process the L3TSE files as defined in the **section 4.2.**	M
4.1	L3TT shall use **Selected.xml** to get the list of applicable Test Cases and their related applicable Pass Criteria.	M
4.2	L3TT shall display the applicable Test Cases and their related applicable Pass Criteria to the user.	M

Req	Requirement	M/O/C
4.3	L3TT shall use the Pass Criteria definition in **RuleSet.xml** <Check> section to proceed with Pass Criteria resolution. Automatized Pass Criteria resolution processing shall comply with the Tool Pass/Fail Automation Criteria defined in **section 4.3.**	M
4.4	L3TT shall be able to retrieve the required data from the relevant L3 formatted logs (e.g., L3 Card to Terminal log and L3 Online Message log) in order to validate the pass criteria as per **section 4.3**.	M
4.5	L3TT shall display the test results of the applicable Test Cases and their related applicable Pass Criteria.	M
4.6	L3TT shall allow the user to: • Attach Log file(s) (L3 formatted or not) or any attachment(s) to a given test case. • Attach Log file(s) (L3 formatted or not) or any attachment(s) globally to the test session. • Add observation to a given test case.	M
4.7	L3TT shall process the character substitutions as described in **section 4.10** while processing the TSE Files.	M
5.0	L3TT shall export a Test Report TSEZ file compliant with the file definition described in **section 4.4.**	M
5.1	L3TT shall update **TestRunInfo.xml** with the <L3TTFIGVersion> tag filled with the version of the L3 Test Tool Engine (including potential bulletin) used at the time of the test run.	M
5.2	L3TT shall update **TestRun.xml** according the directives described in **section 4.2.3** for the following items including Timestamp: • Test results • Log files list • Attachments list • Observations	M
5.3	L3TT shall archive the TSE file, Log files, attachments at TSEZ file root level.	M
5.4	L3TT shall rename the L3 formatted log files with the prefix defined in **section 4.4.**	M

3.4 L3 CS Requirements

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

Table 3.3: L3 CS Requirements

Req	Requirement	M/O/C
1.0	The following **EMV Commands** as defined in the EMV Specifications version 4.3. and **Participant System-specific Commands** shall be supported (1.1 to 1.10):	
1.1	**Compute Cryptographic Checksum (Mastercard)** If the application needs to respond, the command shall be present in the image definition.	M
1.2	**External Authenticate**	M
1.3	**Generate AC and Get Processing Options** If the application needs to respond, the command shall be present in the image definition.	M
1.4	**Get Challenge and Verify**	M
1.5	**Get Data**	M
1.6	**Get Magstripe Data (JCB Legacy)** If the application needs to respond, the command shall be present in the image definition.	M
1.7	**Internal Authenticate**	M
1.8	**Issuer Script Commands (Application Block, Card Block, PIN Change, PIN Unblock, Put Data)** If status bytes other than '9000' are expected in response to an issuer script command, this shall be specified in the card image.	M
1.9	**Read Record** If the application needs to respond, the command shall be present in the image definition.	M
1.10	**Select Application** If the application needs to respond, the command shall be present in the image definition. The application shall support Select Next and Select for blocked application.	M
Note: Successful support/reply to the commands below may depend on personalization data and the correctness of the command		
2.0	**If a command is defined in the card image, the card simulator shall respond as defined in the card image (which may contradict the way the command is defined in EMV or the relevant PS specification).**	M

<table>
<tr><th>Req</th><th>Requirement</th><th>M/O/C</th></tr>
<tr><td>3.0</td><td>If a command is <u>not</u> defined in the card image or the relevant personalization data is not present, then the card simulator shall respond as described below:</td><td></td></tr>
<tr><td>3.1</td><td>Compute Cryptographic Checksum (Mastercard)
The card simulator shall respond with a not OK SW, e.g., 6985.</td><td>M</td></tr>
<tr><td>3.2</td><td>External Authenticate
Criteria: AIP
If AIP B1b3 = 1 and the ARPC is valid, the card simulator shall respond with a SW 9000.
If AIP B1b3 = 1 and the ARPC is <u>not</u> valid, the card simulator shall respond with a SW 6300 or any other SW except 6985 and 9000.
If AIP B1b3=0, the card simulator shall respond with a not OK SW, e.g., 6985.</td><td>M</td></tr>
<tr><td>3.3</td><td>Generate AC and Get Processing Options
The card simulator shall respond with a not OK SW, e.g., 6985.</td><td>M</td></tr>
<tr><td>3.4</td><td>Get Challenge and Verify
The tool is not required to check the CVM List. Always respond as if Enciphered Offline PIN or Plaintext Offline PIN is supported.</td><td>M</td></tr>
<tr><td>3.5</td><td>Get Data
Get Data shall be at least supported for the tags in the list below. These are the ones required by EMV. For all other tags (for example, PS-specific data objects), a SW indicating a failure, like 6985 is allowed unless they are defined in the <InternalTags> or an answer to a Get Data command. If the same tag is present in an answer to a Get Data command and <InternalTags>, then the tag present in the answer to the Get Data command supersedes the <InternalTags>.
<table>
<tr><th>Tag</th><th>Meaning</th><th>Get Data Response</th><th>Comment</th></tr>
<tr><td>9F13</td><td>Last Online ATC Register</td><td>Return any value below the ATC</td><td></td></tr>
<tr><td>9F17</td><td>PIN Try Counter</td><td>Return current value</td><td>Specify this value in Internal tags to support PIN Try Limit reached test cases.</td></tr>
<tr><td>9F36</td><td>ATC</td><td>Return current value</td><td></td></tr>
<tr><td>9F4F</td><td>Transaction Log Format</td><td>9F2701 9F0206 5F2A02 9A03 9F3602 9F5206</td><td>In case another response is required, this can be specified in the Internal tags.</td></tr>
</table></td><td>M</td></tr>
<tr><td>3.6</td><td>Get Magstripe Data (JCB Legacy)
The card simulator shall respond with a not OK SW, e.g., 6985.</td><td>M</td></tr>
</table>

Req	Requirement	M/O/C
3.7	**Internal Authenticate** Criteria: AIP If AIP B1b6 = 1, the card simulator shall respond with a SW 9000. If AIP B1b6 = 0, the card simulator shall respond with a not OK SW, e.g., 6985.	M
3.8	**Issuer Script Commands (Application Block, Application Unblock, Card Block, PIN Change, PIN Unblock, Put Data)** If the application implements the MAC validation, the appropriate SW defined by the PS shall be returned for an invalid MAC.	M
3.9	**Read Record** The card simulator shall respond with a not OK SW, e.g., 6985.	M
4.0	The application **shall support** the following requirements:	
4.1	Correct dynamic ODA handling in order to work with real life terminals.	M
4.2	Application Cryptogram (AC) calculation.	M
4.3	ARPC validation if emvcard.auth() is used.	C
4.4	Computation of Cryptographic Checksum (Mastercard) and dynamic Magstripe for other Participant Systems.	M
4.5	Generation of Card to Terminal (CtT) logs according the format definition in Section 4.5. The application shall export CtT logs in compliance with the applicable format defined by EMVCo.	M
4.6	Implementation of the pseudo-functions defined in the latest version of the *EMV L3 Pseudo-Function Definitions for Test Card Images*.	M
5.0	**The following requirements are NOT mandated, although they may be required by one or more Participant Systems:**	
5.1	MAC validation at issuer scripts.	O
5.2	Offline Enciphered PIN validation.	O
5.3	Card Risk Management function. A terminal test process should not be interested in internal card logic. Therefore, the CID returned by the Card will be specified in the card image, either with a hardcoded value or using one of the pseudo functions.	O
5.4	Correct CVR bit setting. There is no real need for this from an EMVCo Level 3 perspective. Test tool vendors are free to either compute a correct CVR or take a fixed dummy value. The same holds for Counters and Limits.	O

Req	Requirement	M/O/C
5.5	Previous Transaction History.	O
5.6	Support of padding (using ‘00’ bytes) in constructed data objects, including in file records, as specified in EMV Spec Update No. 69.	O
5.7	Support for empty file records.	O
6.0	The ATR value is out-of-scope of L3 testing. It shall be chosen by the tool vendor and must comply with the EMV specifications.	M
7.0	EMV Kernel 8 requirements:	M
7.1	• Support the pseudo functions described in section 3 of the *EMV L3 Pseudo-Function Definitions for Test Card Images* version 1.6 (or higher) document.	M
7.2	• Support the Blinded Diffie-Hellman (BDH) Key Agreement including Elliptic Curve Cryptography (ECC) compliant to the latest version of the EMV Kernel 8 specification.	M
7.3	• When Card Response with Tag ID="DA" is used, the Card Simulator shall encrypt the data using Confidentiality Session Key (SKC). However, the card to terminal log will display the data in clear. The decrypted DA value shall be included in the value of tag <response> of the Card Terminal log starting with tag 70.	M

<table>
<tr><th>Req</th><th>Requirement</th><th>M/O/C</th></tr>
<tr><td>7.4</td><td>
A L3 CS shall be able to identify that they are going down the Kernel 8 path by checking that byte 1 of the Kernel Qualifier (tag '9F2B') version is different from '00'.
<u>Technical implementation details:</u>

If tag '9F2B' is requested in PDOL and byte 1 of tag '9F2B' sent in GPO command is not equal to '00', then Kernel 8 is detected and the L3 CS will select the first most restrictive matching <TerminalRequest> tag which must include an attribute KernelID=08 (if available). An xml test card image would not be able to use the KernelID attribute in a <TerminalRequest> tag prior to the GPO as the L3 CS would not have access to the Kernel Qualifier requested in the PDOL. The table below includes an example.

☑ selected by L3 CS
☒ not selected by L3 CS

<table>
<tr><td>Note: the <TerminalRequest> tags listed in the example below are listed by order in an xml test card supporting both Kernel 8 path and non Kernel 8 path. In this example, it is assumed that the PDOL value starts with '9F2B08', however, tag '9F2B08' can be located at any position in the PDOL.</td><td>If Kernel Qualifier (tag '9F2B') byte 1 = 00, then</td><td>If Kernel Qualifier (tag '9F2B') byte 1 != 00, then</td></tr>
<tr><td><TerminalRequest name="GPO" cmd="80" ins="A8" p1="00" p2 = "00" cmdData="83??0100?*"></td><td>☒</td><td>☒</td></tr>
<tr><td><TerminalRequest name="GPO" cmd="80" ins="A8" p1="00" p2 = "00" cmdData="?????*" KernelID="08"></td><td>☒</td><td>☑</td></tr>
<tr><td><TerminalRequest name="GPO" cmd="80" ins="A8" p1="00" p2 = "00" cmdData="?????*"></td><td>☑</td><td>☒</td></tr>
<tr><td><TerminalRequest name="GenAC" cmd="80" ins="AE" instance="1" cmdData="?*"></td><td>☑</td><td>☒</td></tr>
<tr><td><TerminalRequest name="GenAC" cmd="80" ins="AE" instance="1" cmdData="?*" KernelID="08"></td><td>☒</td><td>☑</td></tr>
</table>
</td><td>M</td></tr>
<tr><td>8.0</td><td>For all other commands not defined above or in the card image, a SW indicating a failure (such as '6985', "command not supported") shall be returned.</td><td>M</td></tr>
</table>

4 EMVCo L3 Detailed Formats

This section describes the details of the different formats of each component.

General requirements:

- The file extension and file name shall not be case sensitive.
- EMVCo reserves the right to add an xml tag. To support backwards compatibility, the test tool shall be able to import a .tse file which, for example, may include additional data. However, the exported .tse file shall be compliant with the applicable version of the specification that was used to qualify the test tool.
- As per the XML syntax described in the w3schools, XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Opening and closing tags shall be written with the same case (e.g., <message>This is correct</message>).
- An XML Schema Definition (XSD) file is available from EMVCo for each xml file defined below. They are only used for qualification purposes by EMVCo. Please contact EMVCo via the query system to request for the files.
- All attributes and defined values used in the files are case-sensitive unless otherwise stated in this document or in the XSD.
- The order of the occurrence of the XML tags shall not be modified unless otherwise stated in this document or in the XSD (i.e., <xs:all>).
- Leading and trailing whitespace for a tag's value must be ignored during processing.
- The value may be empty as defined in the XSD.

For presence data, the following notation is used:

- M: Mandatory – shall always be present.
- C: Conditional – shall be present unless the condition is not met.
- O: Optional – may or may not be present.

For occurrence data, the following notation is used:

- 1..1: Element is mandatory and can only occur once.
- 0..1: Element is optional and if present, can only occur once.
- 1..n: Element is mandatory and can occur more than once.
- 0..n: Element is optional and can occur more than once.

Note: The occurrence rule is bound by its parent element. If the parent element is not present or empty, the occurrence rule for the child elements will not apply.

4.1 TSE Test Set Files

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

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. In the example below, the terminal integration context value is equal to 0111.

Table 4.1: Bitmap position

Bitmap position	Bit 4 – most significant bit	Bit 3	Bit 2	Bit 1 – least significant bit
Context expression	Term_type=ATM	Term_type=AttendedPOS	Interface=Contactless	Interface=Contact
Context Statement	False	True	True	True

The evaluation of context expressions shall not be case sensitive - e.g., "Term_type=Attended POS" is the same as "term_Type=attended pos".

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' (b4 b3 b2 b1).

The first context expression equals the most significant bit in the mask. The logic also applies to Mask1 and Mask2 (both values are defined below).

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

Table 4.2: Masks

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

** It is as 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.

Note: unlike the mask (Mask1 and Mask2) values which must be part of the evaluation performed by the test tool in order to determine the use of a context expression per record, the values populated in the context expression csv columns are used for human readably purposes only.

Processing the TSE Test Set Files

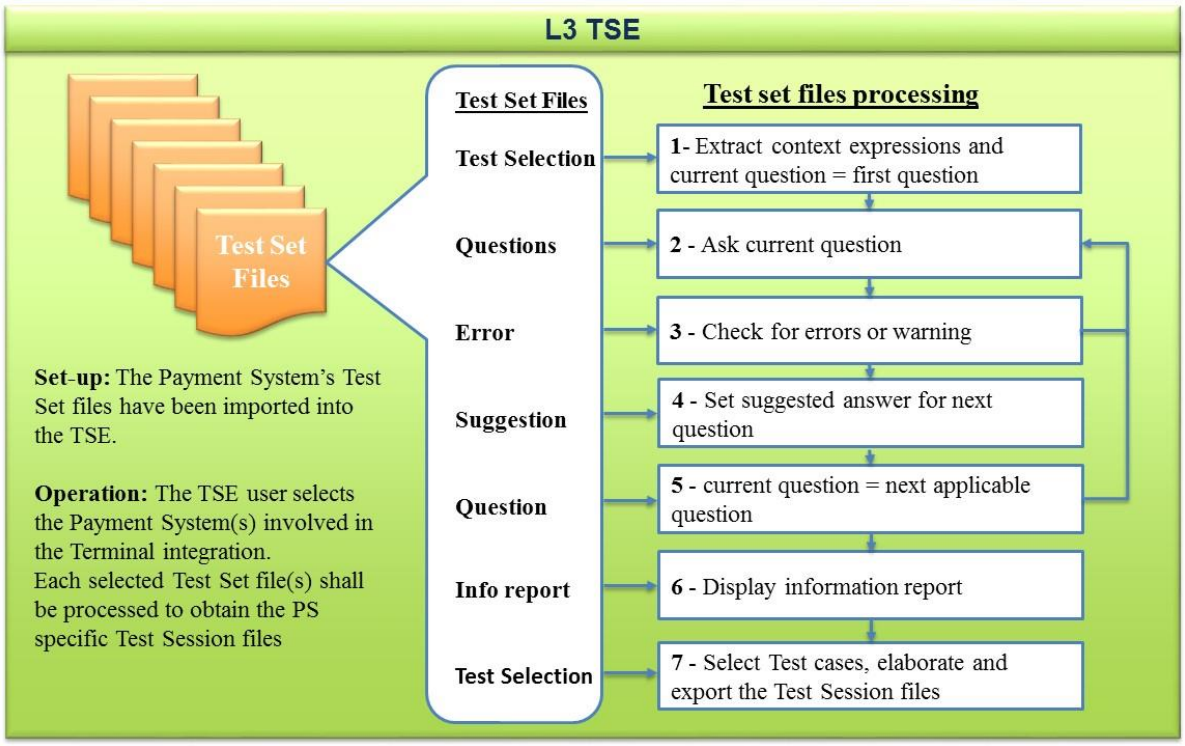
The TSE Test Set files are listed in the table below. They are archived in a **.tsec** file (a renamed .zip file).

Table 4.3: TSE Test Set Files

File Type	Format	Context-related data	Purpose
Test Selection File	csv	Yes	Rules for selecting test cases that are in scope based on the Terminal Integration context
Questions Definition file	csv	Yes	A set of questions that needs to be asked in order to determine the Terminal Integration context
Error Definition file	csv	Yes	Identifies invalid responses to questions and provides appropriate error/warning messages
Suggestions file	csv	Yes	Identifies automatic responses to questions based on the responses to previous questions
Information Report file	csv	Yes	Defines information that should be presented to the tester after the questions have been asked
Card file	csv	No	Definition of cards that are to be used in test cases
Test Case file	csv	No	Definition of all test cases
Pass Criteria file	csv	Yes	Definition of pass criteria requirements for all test cases
Test Reference file	html	No	A readable reference document showing a list of all Test Cases and Test Cards
Manifest file	Text	No	Provides a protected list of all the files output and binds them together into a single data set. This file has extension .tricm.

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.

Figure 4-1: L3 TSE Test Set Files processing



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.
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.

Double quotes cannot appear inside fields unless the field is enclosed with double quotes. If double quotes are used to enclose fields, then a double quote appearing inside a field must be escaped by preceding it with another double quote (e.g., "STRING(""*0200*"")").

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:

"Purpose:
The purpose of the test is to check the ARQC"

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 tag in the above example must either be interpreted as turning on bold text or completely removed from the string. The text "" 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

Text may include HTML references using the standard HTML link syntax of:
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

Table 4.4: File Type Naming convention

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

TSE Test Set files are versioned using:

- P: EMVCo L3 Participant System's Identifier:

Table 4.5: Participant Identifier

Identifier	Participant
00	EMVCo
01	American Express
02	Discover
03	JCB
04	Mastercard
05	UnionPay
06	Visa
07	Verve International
08	eftpos Payments Australia Limited
09 … 99	Reserved for future Participants

- N: Participant System's Test Set Files name, a Participant System proprietary information that may describe the use of the Test Set Files.
- S: Series number, a unique and Participant 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: *00_contact_01_10* identify the TSE Test Set files series n°01 and Build version 1.0 from EMVCo.

Participants shall ensure that a series or a build is generated following a logical numbering convention in order for test tools to not misinterpret the values (e.g., if a Participant provides two TSECs, one filename with a series equal to '01' and the second one with a series equal to '1' then, the test tool could either override the files or create 2 series).

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 questioned 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.

Table 4.6: Test Section File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data
Mask2	
HexMask1	
HexMask2	
TestCase	Unique identifier of the test case. Provided by the Participant System. Each identified Test Case is described in the Test Case Definition file.
Context Expression 1 .. n	One entry per context expression is used to define the Terminal integration context. There is a dedicated field for each bit in the in the bitmap that provides 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 "**unused**"

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 Participant Systems (e.g., some administrative information, some terminal configuration features, etc) within a Project. The TSE shall be able to identify the questions that have already been responded to, retain the related answers, and apply known answers to each Participant System's test session that is part of the same Project. To achieve this requirement, common questions will be identified as those

i. sharing a same criteria name,

ii. sharing the same Name, Type, Allowed and Mode in the 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 Participant Systems are managed separately.

Each record provides the definition of a given question:

Table 4.7: Question Definition File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data.
Mask2	
HexMask1	
HexMask2	
GroupNo	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.
GroupPrompt	Free text used to introduce the Group edit screen to the user.
GroupHelp	Help text made available to the user in HTML format.
GroupMode	Reserved for future use.

Data field	Description and Requirement
Name	Unique name of the question being one of the criteria names used in the context expressions.
Type	Define the type of the question which is related to: • varBoolean - Boolean value. True or False with no preset value (e.g., radio button) unless overwritten by Req 6.6. • varNumber - Numeric value. • varList - mutually exclusive list of strings. • varSet - list where more than more option can be selected. • varString - free text string.
Allowed	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)]
Attributes	Holds a list of attributes, separated by a semi-colon: • Common - defines if the expected answer is common to at least two Participant Systems to avoid redundant question processing. Regression - indicates that changing this question should issue the test cases that are highlighted as regression test cases to be retested.
Prompt	Text to be used to ask the question in HTML format.
Help	Help text made available to the user in HTML format
Mode	Layout indication for the question presentation to the user. Multiple modes may be present, separated by a semi-colon: • **optional** - indicates that the field is optional. If this is not present, then the field should be assumed to be mandatory. • **input_mode=drop_down** - indicates a preference for the user interface to use a dropdown list for options rather than a long list. • **input_mode=drop_down:N** - as above but indicates that the dropdown list should have space for N items. • **input_mode=multiline:N** - 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). • **input_mode=multi_select** - 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. • **input_mode=multi_select:N** - as above but indicates that the lists should have space for at least N items to be displayed. **validation=email** – validates that the entry is an email address. It shall follow the standard email requirements – e.g., test@test.com.

Data field	Description and Requirement
Groups	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 *in* operator.
Context Expression 1 .. n	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

Table 4.8: Error Definition File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data.
Mask2	
HexMask1	
HexMask2	
Name	Internal ID for the error.
ErrorType	Identifies the issue level: • errError - Indicates that this is an error condition that shall be resolved before selecting test cases. errWarning - indicates that this is a warning condition. The user may change a response to a question to resolve it, but shall be allowed to continue to test case selection.
Message	Text error message to be displayed in HTML format.
GotoQuestion	Name of the question that must be changed in order to resolve the error condition.
Context Expression 1 .. n	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.

4.1.5 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:

i. 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 shall be used.
ii. Forced values shall always override suggested values regardless of the order of rules in the file.
iii. Suggestions values for set questions are cumulative - e.g., if there are two active suggestions for a set question then both values shall be proposed.
iv. Forced values for set questions are cumulative - e.g., if there are two active forced values for a set question, then both values shall be forced and the question shall not be displayed.
v. Removals for list and set questions shall be cumulative.
vi. If the question is not in scope and the answer is forced, then that forced answer shall not be applied and the question shall not be displayed.
vii. When Forced values and Suggestions are applicable for set questions, only the forced values shall be applied, and the suggestions shall be ignored.

Each record provides the definition of a given suggestion:

Table 4.9: Suggestion Definition File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data.
Mask2	
HexMask1	
HexMask2	
QuestionName	Unique name of a question in the Question definition file.
SuggestionType	The type of suggestion may have the following values: • force – Contains a list of values that need to be forced for the question • suggest - Contains a suggested list of values for the question

Data field	Description and Requirement
	• remove - Contains a list of values to be removed from the Allowed values list in the Question definition file
Values	List of values to suggest, force or remove in the following format: [value1][value2][value3]
Context Expression 1 .. n	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.

4.1.6 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.

Table 4.10: Information Report File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data.
Mask2	
HexMask1	
HexMask2	
HeadingLevel	Level from 1 to 5.
Name	Internal ID for the Information entry.
SectionHeader	Heading text of the heading level in HTML format.
Information	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.
Context Expression 1 .. n	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:

Table 4.11: Card File Format

Data field	Description and Requirement
ID	Unique card Identifier. Provided by the Participant System.
Name	Friendly name of the card.
Description	Text description of the card technical details.
History	• Major - Integer - major version number incremented with each new major version. • Minor - Integer - minor version incremented with each new minor version. • Description - this is a textual description of the change made. The version numbers are particularly significant to identity the test(s) that should be performed again in case of major change(s).
Tags	Detail a "tag" together with its associated value: • Tag - Name of the tag - e.g., "PIN","PAN","Brand", etc. • Value - Value of the tag - e.g., "4321","5432-0000-0000-0000","PSx Brand". 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". This is represented in the following way: [Tag=PAN:Value=nnnnnnnnnnnnnnn];[Tag=Brand:Value=PSx];[Tag=PIN:Value=4315]

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:

Table 4.12: Test Case File Format

Data field	Description and Requirement
Name	Unique name that corresponds to a test case name in the Test Selection file.
History	• Major - Integer - major version number incremented with each new major version.

Data field	Description and Requirement
	• Minor - Integer - minor version incremented with each new minor version. Description - this is a textual description of the change made.
Cards	Unique card identifier that corresponds to the card identifier in the Card File and optionally include the version of the test card which may be defined in the test card image <header> (if present, the version of the test card is not included as part of the unique card identifier). A Tester may have the choice to select one card in a list: [Card=<test card1>];[Card=<test card2>:Version=<major_version.minor_version>] (e.g., [Card=EMVCoTestCard1];[Card=EMVCoTestCard2:Version=2.2]). When using multiple cards: • Several cards can be used in the same test case. • There is currently no automated card image selection management in the FIG. Manual card image selection may be based on user actions.
Objective	Text description of the test objective in HTML format.
Configuration	Text description of the configuration that is required before test execution.
Applicable	Text description indicating when the test case is applicable in HTML format.
Notes	Text description of notes relates to the test.
Context	For reference only.
Actions	Text description of the action(s) that the tester shall take Format: [Action=<action1>];[Action=<action2>];[Action=<action*n*>].
Requirements	Details the documents or sections of document that is relevant for the test being performed: • Document – Identify the document. • Section – Identify the section in the document. Format: [Document=<document1>:Section=<section1>];[Document=<document 2>:Section=<section2>]
Attributes	Hold a list of attributes for possible filtering purpose Format: [attribute_1];[attribute_2];[attribute_n]. 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.
isRegression	Indicate if the test is for regression testing purpose: Yes – The test should be repeated when the user changes the test selection question(s) after partial or full test completion.

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

Table 4.13: Pass Criteria File Format

Data field	Description and Requirement
Mask1	See Applicability of Context-related Data.
Mask2	
HexMask1	
HexMask2	
Test	Unique test case identifier the check is linked to.
Commentary	Free text describing the purpose of the check.
CheckNumber	The definition is provided in section 4.3.
StepOfCheck	
IsMandatory	
DataItem	
DataFormat	
DataItemName	
Operator	
Value	

Data field	Description and Requirement
ValueItemName	
ActionIfTrue	
ActionIfFalse	
Context Expression 1 .. n	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.

4.1.10 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). It shall not include any duplicate lines.

```
PSI             =00
Series          =0
Build           =140
Name            =DemoPlan
Mode            =EMVCoL3
Version         =1.2
Selection_File  =00_DemoPlan_0_140_selection.csv:       LEy6saV522S2l8h+jwPxcKYW7I8=:
Scenario_File   =00_DemoPlan_0_140_cases.csv:           MXbW1fj+qZYAUa1fvJbOohVjGxs=:
Question_File   =00_DemoPlan_0_140_questions.csv:       j13wfAVBV3rr/p+0A3ZVTSqWNC0=:
Error_File      =00_DemoPlan_0_140_error.csv:           V9bh7yrF8s0KMgStFJ0kbmenxbo=:
Suggest_File    =00_DemoPlan_0_140_suggest.csv:         nGgXbqGRWySNmyC2x9j+LmRQ8bI=:
Pass_Criteria_File =00_DemoPlan_0_140_pass_criteria.csv: QRPu96j3zODQvlxSAGITUObh7WY=:
Card_File       =00_DemoPlan_0_140_card.csv:            P92BJUcBHRqEinwxN50DvGNwUT4=:
Information_File =00_DemoPlan_0_140_info.csv:           KmfR/M5S6TCAc2svwvZ4t963NiM=:
Test_Reference  =00_DemoPlan_0_140_test_reference.html: eI4Pj5tA0EHL8XHYOtbbseLe7qg=:
Signature       =vzFajPLsUNk=
```

- **PSI:** refer to Naming conventions section in 4.1.1 for further details.
- **Series:** Refer to Naming conventions section in 4.1.1 for further details.
- **Build:** Refer to Naming conventions section in 4.1.1 for further details.
- **Name:** Refer to Naming conventions section in 4.1.1 for further details.
- **Mode:** Main usage of the Test Set file. Predefined value is “EMVCoL3”.
- **Version:** EMV L3 Framework - Implementation Guide (FIG) version used to generate the TSEC-package. The presence of this field in the manifest is optional (if Participant System supports it).

  X.Y.zzz where X is a decimal number which represents a major version of the FIG, Y is a decimal number which represents a minor version of the FIG and zzz is an optional value which would represent the latest bulletin decimal number used between 001 and 999 which must be linked to a specific X.Y FIG version (e.g., the Version for this document would be: 1.2 additional examples: 1.3, 1.3.001, 1.3.002, 2.0, etc).

- **Files [9 .. n]:** minimum list of files included in a TSEC-package. Each file in the list is defined by a filename, and corresponding hash value to ensure data integrity:
  - As in the example above a filename is delimited by '=' and ':'. This is always followed by a mandatory hash value that is terminated by a ':'.
  - The hash value contains the SHA-1 hash of the file converted to a Base64 string.

Note: a TSEC-package may optionally include additional files (e.g., Doc_Style=00_DemoPlan_0_140_screenstyle.css:checksum, Release_Note=00_DemoPlan_0_140_releasenotes.txt:checksum, etc.) which may be added to the manifest file. If present, an additional file related line would be included in the manifest file detailing the name, filename and hash (following the format described in table above). The manifest may also contain additional fields not defined in this document. If present, additional fields must be ignored during processing. However additional files and fields are part of the manifest signature calculation when supported.

- **Signature (optional value):** The MAC is calculated as the last block of a CBC single DES encryption of the manifest file excluding end of line characters and excluding the "Signature=MAC" line using a confidential key and IV provided by individual Payment Systems.

  For the MAC calculation purposes:

  a) Any encoding byte order marks at the beginning of the file are ignored.

  b) If the number of input bytes over which the signature is calculated is not a multiple of 8 bytes then padding must be added. The padding type is PKCS7. For example, if one byte of padding is required then it will be the value 0x01. If two bytes of padding are required then the padding will be 0x02 0x02, if three bytes of padding are required the value will be 0x03 0x03 0x03 etc.

  Note: If the signature is present and the signature validation fails, then the tool should display a warning message and continue the execution.

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 and TT 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 – Mandatory - Provides a protected list of all the files output and binds them together into a single data set.

The xml prolog shall be included in each xml file.

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.

The following table indicates which L3 component can read and/or write depending on the TSE Test Sessions Files:

Table 4.14: TSE Test Session Files component rights

	L3 TT		**L3 TSE**	
	Read	**Write**	**Read**	**Write**
TestRunInfo.xml	X	X	X	X
RuleSet.xml	X		X	X
TestRun.xml	X	X	X	X
Selected.xml	X			X

4.2.1 TestRunInfo.xml

Table 4.15: TestRunInfo file Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
`<?xml version="1.0" encoding="utf-8"?>`	`Tag`	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
`<L3tse>`	`Block-start`	M	1..1	XML root node.
`file_version`	`Attribute`	M	1..1	Version of the format definition.
`originator`	`Attribute`	M	1..1	Name of L3TSE tool generating the file.
`.<TestRunInfo>`	`Block-start`	M	1..1	Block which provides information of the test session to the L3TSE for testing session status management purpose.
`..<Context>`	`Block-start`	M	1..1	Single entry.
`...<TrackingNo>`	`Tag`	M	1..1	Unique identifier of the Terminal Integration test session formatted as <Mode from Manifest file>_<PSI>_<timestamp in milliseconds> The format of the timestamp in the TrackingNo shall be defined as follows:YYYYMMDDTHHmissmmmZ (UTC). For example: Year=2021 Month=03 Day=02 T=Time separator Hours=11 Minutes=22 Seconds=33 Milliseconds=444 Z=the zone designator for the zero UTC offset
`...<CloneOf>`	`Tag`	O	0..1	RFU
`...<CopyOf>`	`Tag`	O	0..1	When present holds the TrackingNo of the test run the copy is from.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<MachineID>	Tag	O	0..1	The Machine ID field 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.
...<RuleSetName>	Tag	M	1..1	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. Information from Manifest file.
...<RuleSetSeries>	Tag	M	1..1	
...<RuleSetBuild>	Tag	M	1..1	
...<RuleSetMode>	Tag	M	1..1	Purpose of the Test run (RuleSet) = "EMVCoL3" Terminal Integration. Information from Manifest file.
...<TestReferenceFile>	Tag	O	0..1	Files containing test reference information, help information and CSS style sheets. Stores a link to the filename that contains the information.
...<HelpFile>	Tag	O	0..1	
...<DocStyleFile>	Tag	O	0..1	
...<ScreenStyleFile>	Tag	O	0..1	
...<ChangeFlagDate>	Tag	M	1..1	At package creation, TSE shall set this date to the current date and time.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<Mode>	Tag	M	1..1	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_Info: All questions have been answered and test information (Information Report) is being displayed. • MODE_TestResult: All questions have been answered and test results are being recorded. • MODE_Regression: Questions have been changed and the user should be prompted to clear regression test results. <Mode> flow: • Not all questions are answered -> MODE_Question • All questions are answered and the information report is being displayed-> MODE_Info • Test cases have been selected and are being displayed (test results can be recorded) -> MODE_TestResult • All questions are answered and the user changes the answer to a question with attribute=regression -> MODE_Regression • The status of the regression test cases (test cases marked as “isRegression=Yes”) is reset and the test cases are being displayed -> MODE_TestResult **Note:** the export of a .tse file is required for MODE_Question, MODE_TestResult and MODE_Info but not required for MODE_Regression.
...<FilterActive>	Tag	O	0..1	Tags to keep track of Filtering by the user.
...<FilterOptions>	Tag	O	0..1	
...<SelectedTests>	Tag	O	0..1	User discretionary sub-selection of test cases.
...<L3TSEFIGVersion>	Tag	M	1..1	Version of the L3 Test Selection Engine (including potential bulletin) used at the time of the test run (e.g., 1.1.251, 1.2, 2.0, etc).

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<L3TTFIGVersion>	Tag	C	0..1	**Condition**: if a L3TT is used, then the L3TT will add this tag to the file. Version of the L3 Test Tool Engine (including potential bulletin) used at the time of the test run (e.g., 1.1.251, 1.2, 2.0, etc).
..</Context>	Block-end			
.</TestRunInfo>	Block-end			
</L3tse>	Block-end			

4.2.2 RuleSet.xml

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

- 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.

Table 4.16: RuleSet file Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
`<?xml version="1.0" encoding="utf-8"?>`	`Tag`	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
`<L3tse>`	`Block-start`	M	1..1	XML root node.
`file_version`	`Attribute`	M	1..1	Version of the format definition.
`originator`	`Attribute`	M	1..1	Name of L3TSE tool generating the file.
`.<Questions>`	`Block-start`	M	1..1	A set of questions that needs to be asked to determine the project context. All come from the Question definition file.
`..<Question>`	`Block-start`	M	1..n	Individual entry.
`...<mask1>`	`Tag`	M	1..1	
`...<mask2>`	`Tag`	M	1..1	
`...<groupno>`	`Tag`	M	1..1	
`...<groupprompt>`	`Tag`	M	1..1	
`...<grouphelp>`	`Tag`	M	1..1	
`...<groupmode>`	`Tag`	M	1..1	
`...<name>`	`Tag`	M	1..1	
`...<type>`	`Tag`	M	1..1	

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<allowed>	Tag	M	1..1	
...<attributes>	Tag	M	1..1	
...<prompt>	Tag	M	1..1	
...<help>	Tag	M	1..1	
...<mode>	Tag	M	1..1	
...<groups>	Tag	M	1..1	
..</Question>	Block-end			
.</Questions>	Block-end			
.<Tests>	Block-start	M	1..1	Definition of all test cases.
..<Test>	Block-start	M	1..n	Individual entry.
...<name>	Tag	M	1..1	
...<history>	Tag	M	1..1	
...<cards>	Tag	M	1..1	
...<objective>	Tag	M	1..1	
...<configuration>	Tag	M	1..1	
...<applicable>	Tag	M	1..1	
...<notes>	Tag	M	1..1	
...<context>	Tag	M	1..1	
...<actions>	Tag	M	1..1	
...<requirements>	Tag	M	1..1	
...<attributes>	Tag	M	1..1	
...<isRegression>	Tag	M	1..1	
..</Test>	Block-end			
.</Tests>	Block-end			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.<Errors>	Block-start	M	1..1	Identifies invalid responses to questions and provides appropriate error/warning messages.
..<Error>	Block-start	O	0..n	Individual entry.
...<mask1>	Tag	M	1..1	
...<mask2>	Tag	M	1..1	
...<name>	Tag	M	1..1	
...<ErrorType>	Tag	M	1..1	
...<message>	Tag	M	1..1	
...<GotoQuestion>	Tag	M	1..1	
..</Error>	Block-end			
.</Errors >	Block-end			
.<Selection>	Block-start	M	1..1	Rules for selecting test cases that are in scope based on the context of the project. Reflect Mask1, Mask2 and TestCase from Selection file.
..<Rule>	Block-start	M	1..n	Individual entry.
...<Mask1>	Tag	M	1..1	
...<Mask2>	Tag	M	1..1	
...<TestCase>	Tag	M	1..1	
..</Rule>	Block-end			
.</Selection>	Block-end			
.<CardRecords>	Block-start	M	1..1	Definition of cards that are to be used in test cases.
..<card>	Block-start	M	1..n	Individual entry.
...<id >	Tag	M	1..1	
...<name>	Tag	M	1..1	
...<description>	Tag	M	1..1	

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<history>	Tag	M	1..1	
...<tags>	Tag	M	1..1	
..</card>	Block-end			
.</CardRecords>	Block-end			
.<InformationReport>	Block-start	M	1..1	Defines information that should be presented to the tester after the questions have been asked.
..<Message>	Block-start	O	0..n	Individual entry**.**
...<Mask1>	Tag	M	1..1	
...<Mask2>	Tag	M	1..1	
...<HeadingLevel>	Tag	M	1..1	
...<SectionHeader>	Tag	M	1..1	
...<Information>	Tag	M	1..1	
..</Message>	Block-end			
.</InformationReport>	Block-end			
.<Bitmap>	Block-start	M	1..1	List the Criteria names in the right Bitmap order.
..<Bit>	Block-start	M	1..n	Individual entry
...<Criteria>	Tag	M	1..1	
..</Bit>	Block-end			
.</Bitmap>	Block-end			
.<Checks>	Block-start	M	1..1	Definition of pass criteria requirements for all test cases.
..<Check>	Block-start	M	1..n	Individual entry.
...<mask1>	Tag	M	1..1	
...<mask2>	Tag	M	1..1	
...<Test>	Tag	M	1..1	
...<Commentary>	Tag	M	1..1	

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<CheckNumber>	Tag	M	1..1	
...<StepOfCheck>	Tag	M	1..1	
...<IsMandatory>	Tag	M	1..1	
...<DataItem>	Tag	M	1..1	
...<DataFormat>	Tag	M	1..1	
...<DataItemName>	Tag	M	1..1	
...<Operator>	Tag	M	1..1	
...<Value>	Tag	M	1..1	
...<ValueItemName>	Tag	M	1..1	
...<ActionIfTrue>	Tag	M	1..1	
...<ActionIfFalse>	Tag	M	1..1	
..</Check>	Block-end			
.</Checks>	Block-end			
.<Suggestions>	Block-start	M	1..1	Identifies automatic responses to questions based on the responses to previous questions.
..<suggestion>	Block-start	O	0..n	Individual entry**.**
...<mask1>	Tag	M	1..1	
...<mask2>	Tag	M	1..1	
...<QuestionName>	Tag	M	1..1	
...<SuggestionType>	Tag	M	1..1	
...<Values>	Tag	M	1..1	A boolean value in <Values> shall be coded in lower case (e.g., if the Values in the suggest.csv file indicates [True] or [False], then it shall be converted to [true] or [false] in the <Values> tag).
..</suggestion>	Block-end			
.</Suggestions>	Block-end			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
</L3tse>	Block-end			

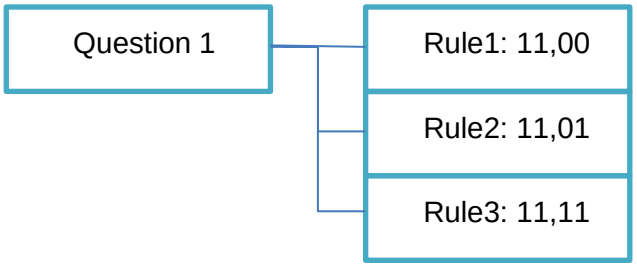
Processing Records with multiple mask values

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:

Table 4.17: Processing Records with multiple mask values

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 tables below describe how the initial TestRun.xml can be created and updated depending on the L3 Component used. It should only reflect the applicable test cases and pass criteria. Inapplicable ones should be excluded from the XML.

The L3 TSE component will create the file in the following way:

Table 4.18: TestRun file Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
<?xml version="1.0" encoding="utf-8"?>	Tag	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
<L3tse>	Block-start	M	1..1	XML root node.
file_version	Attribute	M	1..1	Version of the format definition.
originator	Attribute	M	1..1	Name of L3 TSE tool creating the file.
.<Observations>	Block-start	M	1..1	Give the test observation for a given Test Case.
..<Test>	Block-start	O	0..n	Individual test observation.
...<TestName>	Tag	M	1..1	From RuleSet XML: <Tests><Test><name>.
...<MajorVersion>	Tag	M	1..1	From RuleSet XML: <Tests><Test><history> Major tag value.
...<Observation>	Tag	M	1..1	Create and leave empty (e.g., <Observations/>).
...<ReviewComments>	Tag	M	1..1	
...<Updated>	Tag	M	1..1	Timestamp.
..</Test>	Block-end			
.</Observations>	Block-end			
.<Logfiles>	Block-start	M	1..1	List the log files that have been recorded during the test session.
..<Entry>	Block-start	O	0..n	Individual entry.
...<Test>	Tag	M	1..1	From RuleSet XML: <Tests><Test><name>.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<Files>	Tag	M	1..1	Create and leave <Files> tag empty (e.g., </Files>).
...<Updated>	Tag	M	1..1	Creation Timestamp.
..</Entry>	Block-end			
.</Logfiles>	Block-end			
.<Attachments>	Block-start	M	1..1	List the reference documents that could be helpful for the test session.
..<Entry>	Block-start	O	0..n	One entry.
...<Files>	Tag	M	1..1	Create and leave <Files> tag empty (e.g., </Files>).
..</Entry>	Block-end			
.</Attachments >	Block-end			
.<Answers>	Block-start	M	1..1	List the answers to the questions from <Questions> section in the RuleSet XML file. Answer(s) are supplied for a given question.:
..<Response>	Block-start	O	0..n	Individual entry. The <Response> block shall only be present for questions that are in scope.
...<Name>	Tag	M	1..1	From RuleSet XML <Questions><Question><name>.
...<Answer>	Tag	M	1..1	The answer given to this question during the L3 TSE process. The following list provides additional requirements for specific cases: • A single value (e.g., from VarSet type question) shall be coded as per the following example: <Answer>[def]</Answer>. • Multiple answers (e.g., from VarSet type question) shall be coded using brackets, as per the following example: <Answer>[def][ghi]</Answer>. • A boolean value in a <Answer> tag shall be coded in lower case (i.e., "true" or "false"). L3TT: Read answer(s) for Pass criteria resolution when required.
...<Updated>	Tag	M	1..1	Timestamp.
..</Response>	Block-end			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.</Answers>	Block-end			
.<Results>	Block-start	M	1..1	Applicable check results of the test session.
..<Test>	Block-start	O	0..n	Provides the test results of the test session.
...<TestName>	Tag	M	1..1	From RuleSet XML: <Tests><Test><name>.
...<MajorVersion>	Tag	M	1..1	From RuleSet XML: <Tests><Test><history> Major tag value.
...<CheckNo>	Tag	M	1..1	From RuleSet XML: <Checks><Check><CheckNumber>.
...<StepNo>	Tag	M	1..1	From RuleSet XML: <Checks><Check><StepOfCheck>.
...<Result>	Tag	M	1..1	"not tested". **Note:** The value in the <Result> tag (i.e., "not tested") is case-sensitive. The value shall be populated in lower case.
...<Updated>	Tag	M	1..1	Timestamp.
..</Test>	Block-end			
.</Results>	Block-end			
</L3tse>	Block-end			

During execution, the L3 TT tool or L3 TSE component (if using manual mode) will update the TestRun.xml file in the following way:

Section	Update
.<Observations>	Provide the tester observations for the test session.
..<Test>	Give the test observation for a given Test Case.
...<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 tool.
...<ReviewComments>	
...<Updated>	Timestamp.
.<Log files>	List the log files that have been recorded during the test session.

Section	Update
..<Entry>	
...<Test>	
...<Files>	Comma separated of quoted Log file name(s) + optional directory path - e.g., c:\Logfiles\Logfile1.txt","c:\Logfiles\Logfile2.pdf","Logfile3.xml". **Note:** <Files> may include additional log file formats.
...<Updated>	Timestamp.
.<Attachments>	List the reference documents that could be helpful for the test session.
..<Entry>	
...<Files>	Comma separated of quoted file name(s) + optional directory path - e.g., "c:\Logfiles\file3.txt","c:\Logfiles\file4.pdf","Logfile3.xml". **Note:** <Files> may include additional log file formats.
.<Answers>	List the answers to the questions in <Questions> section in the RuleSet XML file.
..<Response>	Answer(s) supplied for a given question.
...<Name>	
...<Answer>	
...<Updated>	
.<Results>	Provide the check results of the test session Multiple entries.
..<Test>	Give the result of a given Pass Criteria.
...<TestName>	
...<MajorVersion>	
...<CheckNo>	
...<StepNo>	
...<Result>	"not tested", "fail" or "pass". **Note:** The values in the <Result> tag (i.e., "pass" and "fail") are case-sensitive. The value shall be populated in lower case. **Note:** If a check step is not executed, then the <Result> shall be left to "not tested".
...<Updated>	Update Timestamp.

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.

Table 4.19: Selected file Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
<?xml version="1.0" encoding="utf-8"?>	Tag	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
<L3tse>	Block-start	M	1..1	XML root node.
file_version	Attribute	M	1..1	Version of the format definition.
originator	Attribute	M	1..1	Name of L3TSE tool generating the file.
.<InScope>	Block-start	M	1..n	List of the test cases that TSE has defined in Test Session scope.
..<Test>	Block-start	M	1..1	Individual entry.
...<name>	Tag	M	1..1	From RuleSet XML: <Tests><Test><name>.
...<status>	Tag	M	1..1	"In-Progress".
..</Test>	Block-end			
.</InScope>	Block-end			
.<ActivePassCriteria>	Block-start	M	1..1	List of the pass criteria that TSE has defined in Test Session scope.
..<Step>	Block-start	M	1..n	Individual entry.
...<TestName>	Tag	M	1..1	From RuleSet XML: <Tests><Test><name>.
...<MajorVersion>	Tag	M	1..1	From RuleSet XML: <Tests><Test><history> Major tag value.
...<CheckNo>	Tag	M	1..1	From RuleSet XML: <Checks><Check><CheckNumber>.
...<StepNo>	Tag	M	1..1	From RuleSet XML: <Checks><Check><StepOfCheck>.
..</Step>				
.</ActivePassCriteria>				
</L3tse>	Block-end			

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.

- **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.) 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 Participant System’s test cases, the FIG has defined 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 Participant System.

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

Table 4.20: Tool Pass/Fail Automation Criteria syntax

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 shall 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 shall 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	Refer to Tool Pass/Fail Automation Criteria – DataItem in annex B for details.
DataFormat	This field describes the expected format of the data item and is comprised of a single letter prefix followed by an optional length indicator: - nL = numeric data of length L digits - e.g., "1234" is of format "n4". - hL = hex data of length L nibbles - e.g., "FF" is of format "h2". - aL = alphanumeric data of length L characters - e.g., "hello" is of format "a5". - bL = binary data of length L bits. Variable length data is supported by appending a following "-" and a maximum length. For example: - n1-19 - variable length number between 1 and 19 digits - e.g., a PAN - h4-8 - A hex value of 2 to 4 bytes (4 to 8 nibbles). This field will be empty if the DataItem field is "MANUAL".

Tag in TSE file	Description
DataItemName	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.

Tag in TSE file	Description
Operator	This field contains the operator to be used when comparing the data item with the value field. The following operators may be present: - "=" - test for equality with the value. - "<>" - test for not equals to the value. - ">" - test for greater than the value. - "<" - test for less than the value. - "<=" - test for less than or equals to the value. - ">=" - test for greater than or equals to the value. - "find" - See below. - "exists" - the test passes if the element exists regardless of the value. - "not exists" - the test passes if the element doesn't exist. - "before" - the test passes if the element appears in the log before another element. - "after" - the test passes if the element appears in the log after another element. - "not after" - the test passes if the element is not after another element in the log. - "like" - the test passes if the element is like the value (see below). 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. 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 - e.g., NET.0?00.DE.002 find STRING(PAN Value). The “like” operator is used to compare a data element with a pattern string. At present the pattern string supports two wildcard characters: * – matches zero or more characters ? – matches any single character Typical examples include: 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.

Tag in TSE file	Description
Value	This field holds the value to be compared with the data item using the defined operator. The field uses one of the following formats: - HEX(<hex_value>) - A hexadecimal value - e.g., HEX(9000). - NUMBER(<number>) - A numeric value - e.g., NUMBER(12345). - BINARY(<binary>) - A binary value - e.g., BINARY(101). - STRING("<text>") - A string value - e.g., STRING("EMVCo"). - ITEM(<data_item>) - A data item - e.g., ITEM(APDU.80AE.IFD.BYTE.24-4). - QUESTION(<name>) – <Answer> value to the question <name> located in TestRun file - e. g., <name>=CVM_Limit and <Answer>=3000. - Free Text - A text description of the manual check (only when data item name is MANUAL). **Note:** Except for the BINARY function, the value used in the functions above shall not perform any conversion. **Note:** If comparing two STRINGs, then the comparison shall follow the Compare(String, String) Method logic defined in the Microsoft .NET Core 3.1. 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). In the case where the DataItem is using the LENGTH operator the available options are reduced: - NUMBER (<number>) - 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. - LENGTH(<data_item>) - The length of another data item. - ITEM(<data_item>) - The value of another data item (see note below). **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: - item1 = 0123 - item2 = 2 The following tests should both evaluate to true: - LENGTH(item1) = 4 - The length is compared in numeric digits. - LENGTH(item1) = item2 - The length is compared in bytes.
ValueItemName	In the case where the comparison is with another data item, this field contains the "friendly" name of the other item involved.

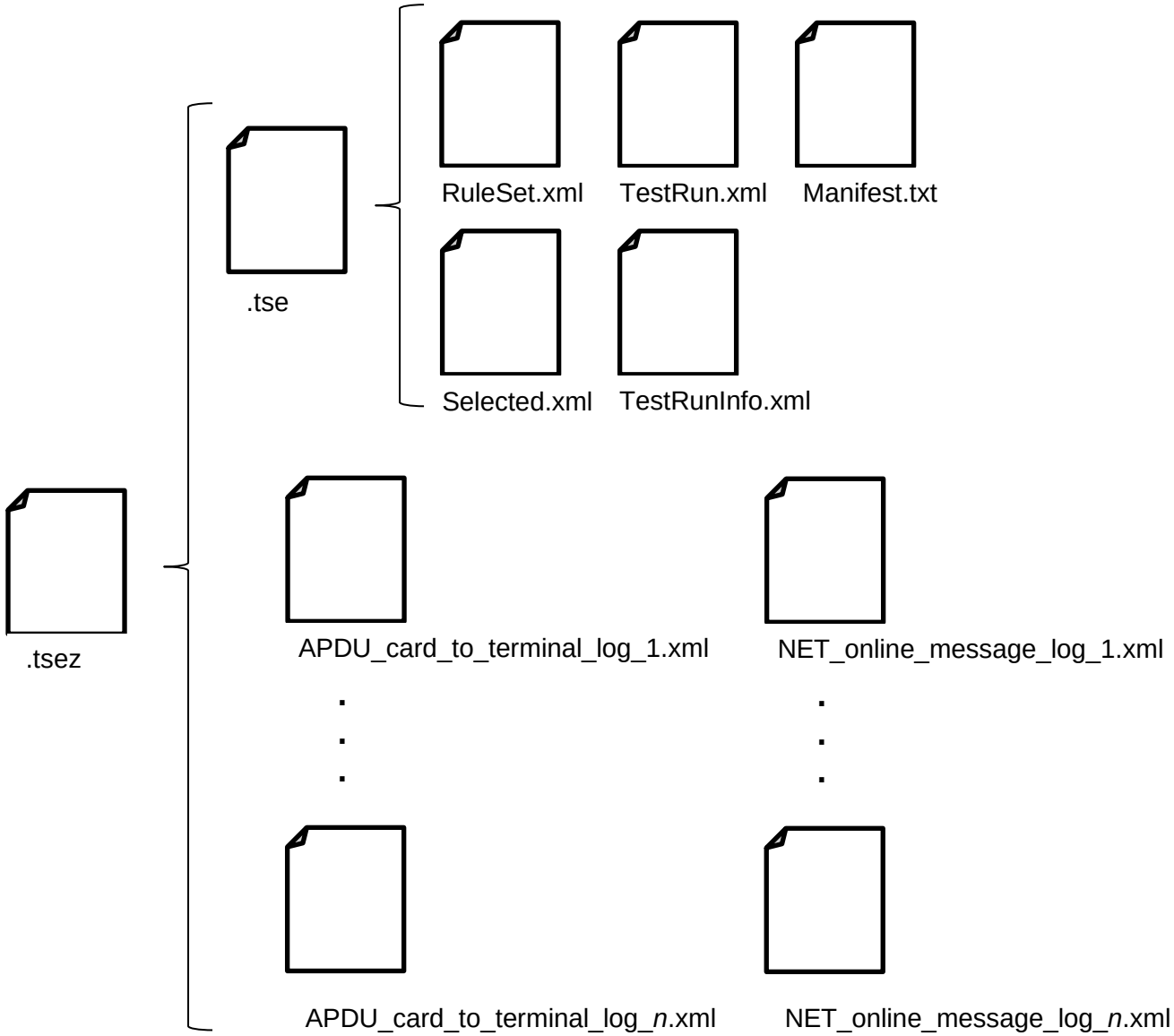
Tag in TSE file	Description
ActionIfTrue	This field indicates what action to take if the result of the comparison is true. The field may contain one of two values: - "PASS" - This indicates that the entire check has passed and there is no need to perform any additional steps for this check. Processing shall continue on the check that has the next highest check number. - <step no> - This indicates that an additional step needs to be performed before the check can be considered as completed. The <step no> indicates that control shall 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.
ActionIfFalse	This field identifies what action to take if the result of the comparison is false. The field may contain one of two values: - "FAIL" - This indicates that the entire check has failed and there is no need to perform any additional steps for this check. Processing shall continue on the check that has the next highest check number. - <step no> - This indicates that an additional step needs to be performed before the check can be considered to have failed. The <step no> indicates that control shall 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.

4.4 Test Report

The Test Report file is an archive file renamed .tsez, structured as shown below and made of:

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

Figure 4-2: TSEZ File Structure Example



To differentiate an online message log from a card to terminal log a prefix shall be provided:

- "NET_" for Online Message logs (e.g., **NET**_Test Case_01.xml).
- "APDU_" for Card to Terminal logs (e.g., **APDU**_Test_Case_01.xml).

4.5 Test Card Image File Format

This part of the guidelines describes the EMVCo Level 3 Card Image definition for Test Card Simulation (L3 Card Simulator- L3CS) using Programmable Cards or Probes.

The Test Card Image shall be machine-readable to facilitate automated implementation by test tool Vendors.

The card image shall serve as input to the EMVCo Level 3 Card Simulator.

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 Introduction

The fundamental encoding of the Test Card Image File is XML with the following blocks:

1. Header information.
2. Features: Card Characteristics & Special behavior that cannot be deduced from the perso content, or that should be highlighted for usability reasons. This might include the cryptography used, e.g., a PS spec and version. However, defining the CVN may be sufficient if not needed by the Participant Systems.
3. Multiple applications on one card are allowed, including in the (P)PSE.
4. Multiple interfaces on one card are allowed.
5. Cryptographic information.
6. MagStripe data (if any).

4.5.2 Test Card Image Format Requirements

Table 4.21: Test Card Image Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
<?xml version="1.0" encoding="utf-8" ?>	Tag	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
<EMVCoL3CardImage>	Block-start	M	1..1	XML root node.
formatVersion	Attribute	M	1..1	Version of the XSD format definition. This shall have a format such as “1.2”, "13.1" etc., with one digit to the right of the dot.
.<Header>	Block-start	M	1..1	Block which specifies global information about Card Image.
..<CardId>	Tag	M	1..1	Unique Identification value, non-empty string.
..<CardVersion>	Tag	M	1..1	Version of the card - e.g., "1.1", non-empty string.
..<Author>	Tag	M	1..1	Participant system name (Free text).
..<Date-Time>	Tag	M	1..1	The date and time when the file is generated. Expressed in UTC.
..<Description>	Tag	M	1..1	Free text describing L3 Card Image usage.
..<L3FIGVersion>	Tag	C	0..1	L3 FIG document version following the naming convention described in section 4.1.11 Manifest File Format, if supported by the PS.
..<L3PFVersion>	Tag	C	0..1	L3 Pseudo functions document version following the naming convention described in section 4.1.11 Manifest File Format, if supported by the PS.
.</Header>	Block-end			
.<Features>	Block-start	O	0..1	Block which specifies Participant System dedicated information.
..<PaymentSystem>	Tag	O	0..1	Participant System name.
..<PaymentSystemSpecificData>	Tag	O	0..1	Participant System might need additional information that is coded as keywords and values to allow flexibility and avoid any change to the framework.
..keyword	Attribute	M	1..1	e.g., keyword="Crypto".
..Value	Attribute	M	1..1	e.g., value="M/Chip4".
.</Features>	Block-end			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.<Contact> or <Contactless>	Block-start	O	0..2	Block which specifies zero or more contact or contactless application(s). Dual Interface card used in Terminal testing can be specified in a single XML file.
..<Application>	Block-start	O	0..n	An application is made of one or more terminal request(s).
..AID	Attribute	M	1..1	Application Identifier - e.g., AID = "32 50 41 59 2E 53 59 53 2E 44 44 46 30 31". Hexadecimal tag or attribute data can be coded with or without spaces.
..cryptoId	Attribute	O	0..1	Optional attribute cryptoId refers to keys in crypto block below. If no cryptoId is specified, then the application uses the first crypto block specified in the definition.
...<TerminalRequest>	Block-start	M	1..n	A terminal request holds one card response. Logic is driven by the combination of cmd-ins-p1-p2-cmdData. Values not present are wildcards.
...name	Attribute	M	1..1	Name of the terminal request - e.g., name="Select". The attribute name is only used for readability purposes.
...cmd	Attribute	O	0..1	APDU command class - e.g., cmd="00".
...ins	Attribute	O	0..1	APDU command instruction - e.g., ins="A4".
...P1	Attribute	O	0..1	APDU command parameter 1 - e.g., p1="04".
...P2	Attribute	O	0..1	APDU command parameter 2 - e.g., p2="00".
...sfi	Attribute	O	0..1	Short file identifier - e.g., sfi="01".
...record	Attribute	O	0..1	Record number - e.g., record="01".
...instance	Attribute	O	0..1	Instance number of the terminal request - e.g., instance="1".
...cmdData	Attribute	O	0..1	Data received from the terminal - e.g., cmdData="830D0000000000000000??????????". - ?* indicates an unspecified number of free-to-choose bytes. - ?? indicates 1 free-to-choose byte. **Note**: data may be expressed at bit level using single quotes, e.g., “04” can be expressed as ‘00000100’.
...kernelID	Attribute	O	0..1	2-digit value indicating the Kernel used (e.g., KernelID=08 would refer to Kernel 8).

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....<CardResponse>	Block-start	M	1..1	The card response to the terminal request.
....sw	Attribute	O	0..1	Status word - e.g., sw="9000". If SW 9000 is expected, there is no need to specify this attribute. For a blocked card the sw responses to all selects should indicate 6A81.
....state	Attribute	O	0..1	Current state of the card application - e.g., state="STATE_SELECTAPP". For future use.
....nextstate	Attribute	O	0..1	Next state of the card application - e.g., nextstate="STATE_GPO". For future use.
.....<Tag>	Tag	O	0..n	Zero or more Tag that could be a Template e.g ID="6F" name="FCI Template" or a single value - e.g., ID="94" name="Application File Locator (AFL)".
.....ID	Attribute	M	1..1	Unique identifier - e.g., ID="84".
.....name	Attribute	O	0..1	Name of the Tag - e.g., name="DF Name".
.....format	Attribute	O	0..1	Define the format of the Tag value - e.g., format="ISO-8859-1". Default format: HEX.
.....tagpresent	Attribute	C	0..1	tagpresent signals whether the tag shall be included (TRUE) or not (FALSE). In case of tag ID 80 (Template Format 1) the tag and the length shall not be present by adding tagpresent="FALSE" taglengthformat="0". Required for format 1 (defined in EMV Book 3, Data Elements Dictionary).
.....taglengthformat	Attribute	C	0..1	taglengthformat=: • "0" (length not present). • "" (minimal length specifier). • "1" (=> tag length coded as 81xx). • "2" (=> tag length coded as 82xxxx). • etc. Required for format 1.
.....cvn	Attribute	O	0..1	Cryptogram Version Number
.....</Tag>	Tag-end			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....</CardResponse>	Block-end			
...</TerminalRequest>	Block-end			
...<InternalTags>	Block-start	O	0..1	Zero or more tags that can be retrieved by the terminal using GetData .
....<Tag>	Tag	O	0..n	Single value Tag - e.g., for contact cards <Tag ID="9F17" name="PIN Try Limit">00</Tag> if a PTL exceeded test is required.
....ID	Attribute	M	1..1	Unique identifier - e.g., ID="84".
....name	Attribute	O	0..1	Name of the Tag - e.g., name="DF Name".
....format	Attribute	O	0..1	Define the format of the Tag value - e.g., format="ISO-8859-1". Default format is HEX.
....</Tag>	Tag-end			
...</InternalTags>	Block-end			
..</Application>	Block-end			
.</Contact> or </Contactless>	Block-end			
.<Crypto>	Block-start	O	0..n	Block which specifies zero or more Crypto blocks.
.CryptoId	Attribute	O	0..1	Default Crypto block if the CryptoId attribute is not present.
..<PIN>	Tag	O	0..1	Zero or one offline PIN value (if supported by the CVM list) - e.g., "1234".
..<UN>	Tag	O	0..1	Zero or one Unpredictable Number in hexadecimal format e.g., "A3FF56C090D3E921".
..<SymmetricKeys>	Block	O	0..1	Zero or one block of symmetric keys.
...<Key>	Tag	O	0..n	Zero or more symmetric keys.
...name	Attribute	O	0..1	Using name values here, every Participant System is free to add its own keys if needed.
...</Key>	Tag-end			
..</SymmetricKeys>	Tag-end			
..<RSAKeys>	Block-start	O	0..1	Zero or one set of RSA keys.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<CA>, <Issuer>, <ICC> or <PINEncipherment>	Block-start	O	0..n	Zero or several keys among the following RSA keys: <CA>, <Issuer>, <ICC>, <PINEncipherment>. If not present the PIN Encipherment keys are assumed to be equal to the ICC keys.
....<Modulus>	Tag	O	0..1	
....<Exponent>	Tag	O	0..1	
....<PrivateExponent>	Tag	O	0..1	
....<P>	Tag	O	0..1	
....<Q>	Tag	O	0..1	
....<Q-1ModP>	Tag	O	0..1	
....<P-1ModQ>	Tag	O	0..1	
....<ExpModP>	Tag	O	0..1	
....<ExpModQ>	Tag	O	0..1	
...</CA> </Issuer> </ICC> </PINEncipherment>	Block-end			
..</RSAKeys>	Block-end			
..<SM2Keys> and/or <ECCKeys>	Block-start	O	0..2	Elliptic Curve keys.
..<domainId>	Attribute	O		
...<DomainParameters>	Block-start	O	0..8	
...<domainId>	Attribute	O		
....<PrimeP>	Tag	O	0..1	
....<ParameterA>	Tag	O	0..1	
....<ParameterB>	Tag	O	0..1	
....<GeneratorX>	Tag	O	0..1	
....<GeneratorY>	Tag	O	0..1	
....<OrderN>	Tag	O	0..1	

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....<CofactorH>	Tag	O	0..1	
...</DomainParameters>	Block-end			
...<CA>, <Issuers>, <PINEncipherment> and/or <ICC>	Block-start	O	0..4	
....<PublicPointX>	Tag	O	0..1	
....<PublicPointY>	Tag	O	0..1	
....<IdentifierZ>	Tag	O	0..1	
....<PrivateKey>	Tag	O	0..1	
...</CA>, </Issuers>, </PINEncipherment> and/or </ICC>	Block-end			
..</SM2Keys> and/or </ECCKeys>	Block-end			
.</Crypto>	Block-end			
.<MagStripe>	Block-start	O	0..1	Block which specifies zero or one Magstripe data.
..< Track1>	Tag	O	0..1	Track1 data
..format	Attribute	M	1..1	e.g., format="ISO-8859-1"
..</ Track1>	Tag-end			
..< Track2>	Tag	O	0..1	Track2 data
..format	Attribute	M	1..1	e.g., format="ISO-8859-1"
..</ Track2>	Tag-end			
.</MagStripe>	Block-start			
</EMVCoL3CardImage>	Block-end			

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 Introduction

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.

The fundamental encoding of the Card-Terminal Communication Log is XML. 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.

4.6.2 Card Terminal log Format Requirements

Table 4.22: Card Terminal log Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
<?xml version="1.0" encoding="utf-8" ?>	Tag	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
<CardTerminalLog>	Block-start	M	1..1	XML root node.
.<LogDetails>	Block-start	M	1..1	Stores administrative information related to the creation and purpose of the log.
..<LoggingTool>	Block-start	M	1..1	Contains the name and version of the product (L3 Card Simulator) producing the logs.
...<ProductName>	Tag	M	1..1	L3 Card Simulator name.
...<ProductVersion>	Tag	M	1..1	L3 Card Simulator version.
...<L3FIGVersion>	Tag	M	1..1	Version of the EMV L3 Testing Framework Implementation Guidelines (optionally including bulletins) used by the L3 CS component at the time of the test run, following the naming convention described in section 4.1.11 Manifest File Format.
...<L3PFVersion>	Tag	M	1..1	Version of the EMV L3 Testing Framework Pseudo Functions Definitions for Card Images used by the L3 CS component at the time of the test run, following the naming convention described in section 4.1.11 Manifest File Format.
..</LoggingTool>	Block-end			
..<Reference>	Tag	M	1..1	Test case reference number and project information which is generated and provided by the consuming tool.
..<Date-Time>	Tag	M	1..1	Standard UTC timestamp in ISO-8601 format. Example "2022-01-31T08:06:18Z".
..<SchemaSelectionIndex>	Tag	M	1..1	For this version of the Card-Terminal Communication Log, this tag must contain 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.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
..<CardId>	Tag	C	0..1	Unique test Card Identification number used to perform the transaction. Conditional if present in the Card_Image_Definition.xml.
..<CardVersion>	Tag	C	0..1	Test Card Version. Conditional if present in the Card_Image_Definition.xml.
.</LogDetails>	Block-end			
.<Contact>	Block-start	M	1..1	Store actual communication logs of the same format.
..<ConsiderSequences>	Block-start	M	1..1	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.* *If the <Sequence> tag is not present, then all the <CommandResponse>* are in scope.
...<Sequence>	Block-start	O	0..n	Zero or more Sequence blocks.
....<Start>	Tag	M	1..1	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.
....<End>	Tag	M	1..1	The index of last <*CommandResponse*> sub element in the <*Commands*> element to include in the transaction being defined.
...</Sequence>	Block-end			
..</ConsiderSequences>	Block-end			
..<Commands>	Block-start	M	1..1	Used to log the actual commands sent using the technology defined by the parent element.
...<CommandResponse>	Block-start	O	0..n	Zero or more Command Response blocks.
....<Command>	Tag	M	1..1	Either a command APDU or an event literal – e.g., "EVENT_POWER_ON".
....<CommandTimestamp>	Tag	O	0..1	Integer providing the time in milliseconds at which the command was sent or the event occurred.
....<Response>	Tag	M	1..1	Response APDU sent by the card or the response of the card to the event defined by the command.
....<ResponseTimestamp>	Tag	O	0..1	Integer providing the time in milliseconds at which the response was received from the card.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...</CommandResponse>	Block-end			
..</Commands>	Block-end			
.</Contact>	Block-end			
.<Contactless>	Block-start	M	1..1	Store actual communication logs of the same format.
..<ConsiderSequences>	Block-start	M	1..1	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.
...<Sequence>	Block-start	O	0..n	Zero or more Sequence blocks.
....<Start>	Tag	M	1..1	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.
....<End>	Tag	M	1..1	The index of last <*CommandResponse*> sub element in the <*Commands*> element to include in the transaction being defined.
...</Sequence>	Block-end			
..</ConsiderSequences>	Block-end			
..<Commands>	Block-start	M	1..1	Used to log the actual commands sent using the technology defined by the parent element.
...<CommandResponse>	Block-start	O	0..n	Zero or more Command Response blocks.
....<Command>	Tag	M	1..1	Either a command APDU or an event literal – e.g., "EVENT_POWER_ON".
....<CommandTimestamp>	Tag	O	0..1	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.
....<Response>	Tag	M	1..1	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.
....DA	Attribute	C	0..1	An attribute DA shall be added to the <Response> tag if it includes encrypted DA as defined in the contactless Kernel 8 specification.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....<ResponseTimestamp>	Tag	O	0..1	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.
...</CommandResponse>	Block-end			
..</Commands>	Block-end			
.</Contactless>	Block-end			
.<Signature>	Block-start	O	0..1	Encoded according to the XML Digital Signature standard guidelines. Define as ref="ds:Signature" in the .xsd file.
.</Signature>	Block-end			
</CardTerminalLog>	Block-end			

4.6.3 Additional Log Details

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.

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 warm 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.

4.6.4 Signature

In order to verify the integrity of the log, an XML Digital Signature may be attached to each log.

Note: the signature described in this section could also apply to Online Message Logs when present.If a test tool supports Signature, then:

- A receiving tool shall be provided with the RSA public keys in a standard format each associated with an accompanying unique name which shall be associated with a given tool as identified by its <LoggingTool> element.
- The <KeyName> tag shall be used by the sending tool and shall 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.

4.7 Online Message log Format

This online message log format provides authorisation request and response details between Acceptance Devices under test and the online System, this section only defines the structure of the Online Message Format.

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.. 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.

4.7.2 Online Message log Format Requirements

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. EMVCoL3-OMF shall support the breakdown of raw messages into individual fields and subfields. 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 (optional) – a signature block which digitally signs the contents of the EMVCoL3-OMF file.

Table 4.23: Online Message log Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
<?xml version="1.0" encoding="utf-8"?>	Tag	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
<EMVCoL3OnlineMessageFormat>	Block-start	M	1..1	XML root node.
.<LogDetails>	Block-start	M	1..1	Block which specifies the application which created the online message file, the file format version and comments.
..<Date-Time>	Tag	M	1..1	The date and time when the file was generated. Expressed in UTC.
..<LoggingTool>	Block	M	1..1	Block which specifies the application which created the online message log.
...<ProductName>	Tag	M	1..1	The name of the application which created the online message log.
...<ProductVersion>	Tag	M	1..1	The version of the application which created the online message log.
..</LoggingTool>	Block-end	M		
..<SchemaSelectionIndex>	Tag	M	1..1	The version number of the online message log format used. The version of the standard documented here is "1.0".
..<L3OMLVersion>	Tag	M	1..1	This tag contains the version of the FIG used to generate the OML at the time of the test run.
..<Reference>	Tag	M	1..1	Any useful reference information for the data contained within the file.
.</LogDetails>	Block-end	M		
.<ConnectionList>	Block-start	M	1..1	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	1..n	Block which specifies information concerning one end of a point-to-point connection.
..ID	Attribute	M	1..1	A unique symbolic name used to refer to this connection.
...<Protocol>	Block-start	M	1..1	Block which specifies the protocol operating over the connection.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....<FriendlyName>	Tag	M	1..1	The user-viewable name of the protocol used on this connection
....<SymbolicName>	Tag	M	1..1	The machine-readable name of the protocol used on this connection. The protocol format may be made of the PS name and the protocol name separated by a dot. Examples: • VISA.BASE1 • MC.GCIS
....<VersionInfo>	Tag	M	1..1	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	O	0..1	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	1..1	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	1..1	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	1..1	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 – e.g., "Two bytes, network order" or "Four bytes, PDP-order".
....<Client>	Tag	M	1..1	Specifies whether the connection is operating as a client (true) or a server (false).
....<Format>	Tag	M	1..1	A free-format text string which describes the basic format of the data, e.g., "ASCII", "EBCDIC", "BCD" etc.
...</TCPIPParameters>	Block-end	C		

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<DialupParameters>	Block-start	O	0..1	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	1..1	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	1..1	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	1..1	Specifies whether the connection is operating as a client (true) or a server (false).
....<Format>	Tag	M	1..1	A free-format text string which describes the basic format of the data, e.g., "ASCII", "EBCDIC", "BCD" etc.
...</DialupParameters>	Block-end	O		
..</Connection>	Block-end	M		
.</ConnectionList>	Block-end	M		
.<OnlineMessageList>	Block-start	M	1..1	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	0..n	An application-generated comment concerning any information not explicitly related to a specific online message or data exchange (e.g., 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.
..<OnlineMessage>	Block-start	O	0..n	XML block which specifies a single online message or data exchange from one point to another.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
..Class	Attribute	M	1..1	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. The Class attribute shall use either: • "REQUEST" for request online message • or, "RESPONSE" for response online message as part of the attribute name which shall not be case sensitive.
..Source	Attribute	M	1..1	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	M	1..1	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	M	1..1	The raw, uninterpreted, hexadecimal data bytes which make up the complete online message – e.g., "01F0F03031".
...<MessageGroup>	Tag	O	0..1	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	1..1	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	0..1	The value of the PIN recovered from the online message. This tag should not be present if no PIN was recovered. **Note:** if "PINValue" field is present in the Online message then, the PIN value has been successfully decrypted.
....<PINKey>	Tag	O	0..1	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	0..1	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.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
....<ARQCValidated>	Tag	O	0..1	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	0..1	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	0..1	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	0..1	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	0..n	An application-generated comment related to the online message. This tag can appear zero or more times within the enclosing <MessageInfo> blockt.
....<Date-Time>	Tag	M	1..1	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	1..1	Contains zero or more Field objects.
....<Field>	Block-start	M	1..n	A single field within an online message or data block within an online message exchange.
....ID	Attribute	M	1..1	Attribute which specifies the name of the field. The value shall be unique so that the value specified in the field can be located and accessed by any L3 test Tool or code. It is intended that the value of this attribute is only used internally by the L3 TT to identify the field uniquely. The ID value shall be coded as the Network Message-related “DataItem” value defined in the Check definition (Refer to section 4.3 for details.) excluding the BYTE and BIT notation - e.g., NET.0100.DE.048.SE.022.SF.001

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.....<FriendlyName>	Tag	M	1..1	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	0..1	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	O	0..1	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	O	0..1	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	0..1	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	O	0..1	Attribute which specifies, in hexadecimal, the tag identifier of the EMV data element.
.....Name	Attribute	O	0..1	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	Occurrence	Description and Requirement
.....Format	Attribute	O	0..1	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: • "V" – the value of the EMV data element. • "LV" – the data length identifier and the value of the EMV data element. • "TLV" - the tag identifier, the length and the value of the EMV data element. 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.
.....<FieldType>	Tag	M	1..1	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 – e.g., "ANS..17", "N5" etc.
.....<PrefixBinary>	Tag	O	0..1	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	O	0..1	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	M	1..1	Tag which specifies the raw, hexadecimal data for the field or subfield (example 'F0 F1 F0 F0').
.....<FieldViewable>	Tag	M	1..1	Tag which specifies the user-readable form of the field (using the example from <FieldBinary> the value shall be equal to '0100'). A L3 test tool which accesses the fields within the online message shall use the data specified in this tag to compare it with the Value of a Test Case Pass/Fail Criteria (refer to section 4.3 for details) without applying any conversion.

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.....<ToolComment>	Tag	O	0..n	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'). This tag can appear zero or more times within the enclosing <Field> block.
.....<ToolCommentLevel>	Tag	O	0..n	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. This tag can appear zero or more times within the enclosing <Field> block.
.....<FieldList>	Block-start	O	0..n	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		
....</Field>	Block-end	M		
...</FieldList>	Block-end	M		
..</OnlineMessage>	Block-end	M		
.</OnlineMessageList>	Block-end	M		
.<Signature>	Block-start	O	0..1	Encoded according to the XML Digital Signature standard guidelines. Define as ref="ds:Signature" in the .xsd file.
.</Signature>	Block-end	O		
</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).

4.8 TSE Validation Report Files Format

4.8.1 Introduction

The L3TSE XML Validation Report is used to send the feedback (i.e., pass/fail pass criteria status) of the session validation in an XML file that can be loaded in the L3TSE. That file has a .tser extension.

When the XML Validation Report is loaded in the L3TSE, a Validation Report window shall be displayed with the below information:

- The Tracking Number, the date of the report and the validation status: pass or fail
- A free comment section
- Test Review: an optional section listing the test cases containing Review Comments
- Question Review: an optional section listing the questions that may need to be reviewed

The user can decide to apply the changes to the test session. If a change is applied the TSE test session files shall be updated. When the Validation Report is applied, the L3TSE shall apply all the updates to the L3TSE session files and mark the updated test cases as changed. The Validation Report .tser file shall be added to the L3TSE session as a global attachment. For additional details about the L3TSE XML Validation Report please refer to Section 4.8.2.

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.

4.8.2 TSER File Format Requirements

Table 4.24: TSER File Format

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
`<?xml version="1.0" encoding="utf-8"?>`	`Tag`	M	1..1	xml version and encoding e.g., <?xml version="1.0" encoding="utf-8" standalone="no"?>
`<L3tse>`	`Block-start`	M	1..1	XML root node.
`file_version`	`Attribute`	M	1..1	Version of the format definition.
`originator`	`Attribute`	M	1..1	Name of L3TSE tool generating the file.
`.<ValidationInfo>`	`Block-start`	M	1..1	L3TSE Validation Report file information.
`..<Context>`	`Block-start`	M	1..1	Single entry.
`...<TrackingNo>`	`Tag`	M	1..1	Tracking number of the .tse file that has been reviewed.
`...<OriginalFilename>`	`Tag`	M	1..1	Name of the .tse file that has been reviewed.
`...<RuleSetName>`	`Tag`	M	1..1	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 shall warn the user so that they can decide not to proceed with the automatic processing.
`...<RuleSetSeries>`	`Tag`	M	1..1	
`...<RuleSetBuild>`	`Tag`	M	1..1	
`...<ReviewDate>`	`Tag`	M	1..1	Records the data and time of the validation.
`...<ReviewStatus>`	`Tag`	M	1..1	Shall be set to either "pass" or "fail" depending on whether or not the validation was successful. The L3TSE shall display this information when processing the validation report. **Note:** the values in the <ReviewStatus> tag (i.e., "pass" or "fail") are case-sensitive. The value shall be populated in lower case.
`...<ReviewText>`	`Tag`	O	0..1	May contain an html validation report describing the issues found during testing. L3TSE shall display this information when processing the validation report.
`..</Context>`	`Block-end`			

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
.</ValidationInfo>	Block-end			
.<Observations>	Block-start	M	1..1	When processing validation reports, L3TSE could update the <ReviewComments> information for the test and to make the comments visible on the user interface when the test is opened.
..<Test>	Block-start	M	1..n	Multiple entries.
...<TestName>	Tag	M	1..1	
...<Observation>	Tag	M	1..1	
..</Test>	Block-end			
.</Observations>	Block-end			
.<Answers>	Block-start	M	1..1	The validation report format allows for new answers to be provided for one or more questions. If new answers are provided then L3TSE shall update the necessary question responses and switch to question entry mode. New answers are only relevant when the <ReviewStatus> is "fail".
..<Response>	Block-start	M	1..n	Multiple entries.
...<Name>	Tag	M	1..1	
...<Answer>	Tag	M	1..1	
..</Response >	Block-end			
.</Answers>	Block-end			
.<Results>	Block-start	M	1..1	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. When processing failed validations, L3TSE will provide the option for the user to update the status of test steps to match the entries in the validation report.
..<Test>	Block-start	M	1..n	Multiple entries.
...<TestName>	Tag	M	1..1	

Field	Block/Tag/Attribute	M/O/C	Occurrence	Description and Requirement
...<MajorVersion>	Tag	M	1..1	
...<CheckNo>	Tag	M	1..1	
...<StepNo>	Tag	M	1..1	
...<Result>	Tag	M	1..1	Shall be set to "pass" or "fail" or "not tested".
..</Test>	Block-end			
.</Results>	Block-end			
</L3tse	Block-end			

4.9 Cryptographic Processing

4.9.1 Hash Calculation

For data integrity checking purpose TSE Test Session 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 TT shall warn the user if this hash value is not correct. The intention is that automated tools shall always maintain the correct checksum but if the file is manually changed without updating the checksum then a warning should be displayed.

The control characters are highlighted in black in Table 4.25.

An example of hash calculation is detailed below using the TestRunInfo.xml file:

Table 4.25: Hash Calculation Example

Input value **Note:** in the following example, each line ends with the following control characters: LF, except after the </L3tse> tag which ends with CRLF. CR = Carriage Return LF = Line Feed	<L3tse originator="L3 TSE 3.0.0.120" file_version="1">LF <TestRunInfo>LF <Context>LF <TrackingNo>EMVCoL3_02_20210401T093353441Z</TrackingNo>LF <CloneOf></CloneOf>LF <CopyOf></CopyOf>LF <MachineID></MachineID>LF <RuleSetName>L3 TSE Qualification v60</RuleSetName>LF <RuleSetSeries>60</RuleSetSeries>LF <RuleSetBuild>0</RuleSetBuild>LF <RuleSetMode>EMVCoL3</RuleSetMode>LF <TestReferenceFile></TestReferenceFile>LF <HelpFile></HelpFile>LF <DocStyleFile></DocStyleFile>LF <ScreenStyleFile></ScreenStyleFile>LF <ChangeFlagDate>2021-04-01T10:10:01Z</ChangeFlagDate>LF <Mode>MODE_TestResult</Mode>LF <FilterActive></FilterActive>LF <FilterOptions></FilterOptions>LF <SelectedTests></SelectedTests>LF <L3TSEFIGVersion>1.1.251</L3TSEFIGVersion>LF <L3TTFIGVersion></L3TTFIGVersion>LF </Context>LF </TestRunInfo>LF </L3tse>CRLF

Input value in Hex ASCII: **Note:** The Input value in Hex ASCII shall include all the characters (including control characters and spaces).	3c 4c 33 74 73 65 20 6f 72 69 67 69 6e 61 74 6f 72 3d 22 4c 33 20 54 53 45 20 33 2e 30 2e 30 2e 31 32 30 22 20 66 69 6c 65 5f 76 65 72 73 69 6f 6e 3d 22 31 22 3e 0a 20 20 3c 54 65 73 74 52 75 6e 49 6e 66 6f 3e 0a 20 20 20 20 3c 43 6f 6e 74 65 78 74 3e 0a 20 20 20 20 20 20 3c 54 72 61 63 6b 69 6e 67 4e 6f 3e 45 4d 56 43 6f 4c 33 5f 30 32 5f 32 30 32 31 30 34 30 31 54 30 39 33 33 35 33 34 34 31 5a 3c 2f 54 72 61 63 6b 69 6e 67 4e 6f 3e 0a 20 20 20 20 20 20 3c 43 6c 6f 6e 65 4f 66 3e 3c 2f 43 6c 6f 6e 65 4f 66 3e 0a 20 20 20 20 20 20 3c 43 6f 70 79 4f 66 3e 3c 2f 43 6f 70 79 4f 66 3e 0a 20 20 20 20 20 20 3c 4d 61 63 68 69 6e 65 49 44 3e 3c 2f 4d 61 63 68 69 6e 65 49 44 3e 0a 20 20 20 20 20 20 3c 52 75 6c 65 53 65 74 4e 61 6d 65 3e 4c 33 20 54 53 45 20 51 75 61 6c 69 66 69 63 61 74 69 6f 6e 20 76 36 30 3c 2f 52 75 6c 65 53 65 74 4e 61 6d 65 3e 0a 20 20 20 20 20 20 3c 52 75 6c 65 53 65 74 53 65 72 69 65 73 3e 36 30 3c 2f 52 75 6c 65 53 65 74 53 65 72 69 65 73 3e 0a 20 20 20 20 20 20 3c 52 75 6c 65 53 65 74 42 75 69 6c 64 3e 30 3c 2f 52 75 6c 65 53 65 74 42 75 69 6c 64 3e 0a 20 20 20 20 20 20 3c 52 75 6c 65 53 65 74 4d 6f 64 65 3e 45 4d 56 43 6f 4c 33 3c 2f 52 75 6c 65 53 65 74 4d 6f 64 65 3e 0a 20 20 20 20 20 20 3c 54 65 73 74 52 65 66 65 72 65 6e 63 65 46 69 6c 65 3e 3c 2f 54 65 73 74 52 65 66 65 72 65 6e 63 65 46 69 6c 65 3e 0a 20 20 20 20 20 20 3c 48 65 6c 70 46 69 6c 65 3e 3c 2f 48 65 6c 70 46 69 6c 65 3e 0a 20 20 20 20 20 20 3c 44 6f 63 53 74 79 6c 65 46 69 6c 65 3e 3c 2f 44 6f 63 53 74 79 6c 65 46 69 6c 65 3e 0a 20 20 20 20 20 20 3c 53 63 72 65 65 6e 53 74 79 6c 65 46 69 6c 65 3e 3c 2f 53 63 72 65 65 6e 53 74 79 6c 65 46 69 6c 65 3e 0a 20 20 20 20 20 20 3c 43 68 61 6e 67 65 46 6c 61 67 44 61 74 65 3e 32 30 32 31 2d 30 34 2d 30 31 54 31 30 3a 31 30 3a 30 31 5a 3c 2f 43 68 61 6e 67 65 46 6c 61 67 44 61 74 65 3e 0a 20 20 20 20 20 20 3c 4d 6f 64 65 3e 4d 4f 44 45 5f 54 65 73 74 52 65 73 75 6c 74 3c 2f 4d 6f 64 65 3e 0a 20 20 20 20 20 20 3c 46 69 6c 74 65 72 41 63 74 69 76 65 3e 3c 2f 46 69 6c 74 65 72 41 63 74 69 76 65 3e 0a 20 20 20 20 20 20 3c 46 69 6c 74 65 72 4f 70 74 69 6f 6e 73 3e 3c 2f 46 69 6c 74 65 72 4f 70 74 69 6f 6e 73 3e 0a 20 20 20 20 20 20 3c 53 65 6c 65 63 74 65 64 54 65 73 74 73 3e 3c 2f 53 65 6c 65 63 74 65 64 54 65 73 74 73 3e 0a 20 20 20 20 20 20 3c 4c 33 54 53 45 46 49 47 56 65 72 73 69 6f 6e 3e 31 2e 31 2e 32 35 31 3c 2f 4c 33 54 53 45 46 49 47 56 65 72 73 69 6f 6e 3e 0a 20 20 20 20 20 20 3c 4c 33 54 54 46 49 47 56 65 72 73 69 6f 6e 3e 3c 2f 4c 33 54 54 46 49 47 56 65 72 73 69 6f 6e 3e 0a 20 20 20 20 3c 2f 43 6f 6e 74 65 78 74 3e 0a 20 20 3c 2f 54 65 73 74 52 75 6e 49 6e 66 6f 3e 0a 3c 2f 4c 33 74 73 65 3e 0d 0a

SHA-1 Calculation(Input Value in Hex ASCII)	bf37273c61273c4912a41ad8cea60723ed4d0c72

Base64 Encoding(SHA-1 Calculation(Input Value in Hex ASCII))	vzcnPGEnPEkSpBrYzqYHI+1NDHI=

Updated TestRunInfo.xml including the hash value in the comment. **Note:** in the following example, each line ends with the following characters: LF, except after the </L3tse> tag which ends with CRLF. CR = Carriage Return LF = Line Feed **Note:** when verifying the hash value, the comment line itself and any characters (in red) following the comment line shall be excluded from the Input Value (including any control characters in the comment line and after).	<L3tse originator="L3 TSE 3.0.0.120" file_version="1">LF <TestRunInfo>LF <Context> <TrackingNo>EMVCoL3_02_20210401T093353441Z</TrackingNo>LF <CloneOf></CloneOf>LF <CopyOf></CopyOf>LF <MachineID></MachineID>LF <RuleSetName>L3 TSE Qualification v60</RuleSetName>LF <RuleSetSeries>60</RuleSetSeries>LF <RuleSetBuild>0</RuleSetBuild>LF <RuleSetMode>EMVCoL3</RuleSetMode>LF <TestReferenceFile></TestReferenceFile>LF <HelpFile></HelpFile>LF <DocStyleFile></DocStyleFile>LF <ScreenStyleFile></ScreenStyleFile>LF <ChangeFlagDate>2021-04-01T10:10:01Z</ChangeFlagDate>LF <Mode>MODE_TestResult</Mode>LF <FilterActive></FilterActive>LF <FilterOptions></FilterOptions>LF <SelectedTests></SelectedTests>LF <L3TSEFIGVersion>1.1.251</L3TSEFIGVersion>LF <L3TTFIGVersion></L3TTFIGVersion>LF </Context>LF </TestRunInfo>LF </L3tse>CRLF <!--vzcnPGEnPEkSpBrYzqYHI+1NDHI=-->CRLF

4.10 Character Substitution

4.10.1 CSV and XML Character Substitution

When coding a TSEC, the PSI shall substitute all the characters as defined in the following Table 4.26: CSV Character Substitution, except for the characters part of the format of a field.

Table 4.26: CSV Character Substitution

Text in Test Set CSV file	Text on user interface
" &&"	"&"
" &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".

The character substitutions shall be handled as follows by test tools:

- When creating the RuleSet.xml, the L3 TSE shall substitute all the characters from the TSE Test Set files as defined in Table 4.27: XML Character Substitution, except for the characters part of the format of a field.
- When creating or updating an XML file in a .tse file, the test tool shall substitute all the characters in free text fields (e.g., varString Questions, Files, etc) as defined in Table 4.27: XML Character Substitution. This also applies to values that have been populated using a suggestion.

Table 4.27: XML Character Substitution

Text in Test Set CSV file or User Interface	Text in Test Session XML file
&	&amp;
>	>
<	<
'	'
"	"

- When processing/displaying the information to the user, the test tool shall reverse the substitutions in the following order:
    1. Firstly, reverse the character substitutions as defined in Table 4.27: XML Character Substitution if the information is read from an xml file.
    2. Secondly, reverse the character substitutions as defined in Table 4.26: CSV Character Substitution.
    3. Thirdly, display the information to the user without the formatting characters as defined in this document.

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

The examples described in the table below have been created using the Data field Actions format defined in 4.1.8 - Test Case File Format (i.e., Format: [Action=<action1>];[Action=<action2>];[Action=<action*n*>]). The following symbol ⋆ has been used to replace a space in the following Input and Output columns.

Table 4.28: Character Substitution Examples

	Input	**Output**
When creating the RuleSet.xml.	**From the TSE Test Set files:** [Action=Tap⋆your⋆<B>card</B>⋆⋆&⋆review⋆if⋆&cl⋆⋆&obterminal⋆&cb⋆⋆&eq⋆"approved".];[Action=if⋆failed⋆re-tap.]	[Action=Tap⋆your⋆<B>card</B> ⋆&⋆review⋆if⋆&cl⋆⋆&obterminal⋆&cb⋆⋆&eq⋆"approved".];[Action=if⋆failed⋆re-tap.]
When processing/displaying the information to the user from RuleSet.xml.	**From RuleSet.xml:** [Action=Tap⋆your⋆<B>card</B> ⋆⋆&⋆review⋆if⋆&cl⋆⋆&obterminal⋆&cb⋆⋆&eq⋆"approved".];[Action=if⋆failed⋆re-tap.]	**Firstly:** [Action=Tap⋆your⋆<B>card</B>⋆⋆&⋆review⋆if⋆&cl⋆⋆&obterminal⋆&cb⋆⋆&eq⋆"approved".];[Action=if⋆failed⋆re-tap.]
		Secondly: [Action=Tap⋆your⋆<B>card</B>⋆&⋆review⋆if:⋆[terminal]⋆=⋆"approved".];[Action=if⋆failed⋆re-tap.]
		Thirdly: Tap⋆your⋆**card**⋆&⋆review⋆if:⋆[terminal]⋆=⋆“approved”.⋆ If⋆failed⋆re-tap.
When processing/displaying the information to the user from TSE Test Set Files.	**From the TSE Test Set files:** [Action=Tap⋆your⋆<B>card</B>⋆⋆&⋆review⋆if⋆&cl⋆⋆&obterminal⋆&cb⋆⋆&eq⋆"approved".];[Action=if⋆failed⋆re-tap.]	**Firstly:** [Action=Tap⋆your⋆<B>card</B>⋆&⋆review⋆if:⋆[terminal]⋆=⋆"approved".];[Action=if⋆failed⋆re-tap.]
		Secondly: Tap⋆your⋆**card**⋆&⋆review⋆if:⋆[terminal]⋆=⋆“approved”.⋆ If⋆failed⋆re-tap.
When updating the Answer tag in the TestRun.xml (varString)	**Tester’s input:** A⋆&⋆B:⋆<version⋆1.0>	**In TestRun.xml:** A⋆&⋆B:⋆<version⋆1.0>

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 Participant 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	Binary
C	
CAM	Card Authentication Method
Card Acceptor Terminal	*See "Terminal"*
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: **UAT**, 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.
Component	A technical tool (i.e., L3 Card Simulator (L3 CS), L3 Test Tool (L3 TT) engine or L3 Test Selection Engine (L3 TSE)) that when all three combined enables a user to complete a L3 test session.
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	The JCB contactless IC terminal integration testing process.

Implementation (TCI-CL)	
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
D	
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
E	
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.
F	
FCI	File Control Indicator
G	
GPO	Get Processing Options
H	
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 Participant System for transaction authorization and back to the device. This is a series of message formats, currently implemented by more than one protocol.

I	
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.
ISO	International Organization for Standardization
Issuer	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.
K	
Kernel	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.
L	
Limited Amount Device	An unattended device that has data capture-only capability, and accepts payment for items such as parking garage fees, road tolls, etc.
L3 CTT	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)
L3 LoQ	L3 Letter of Qualification – a formal letter issued to a L3 Test Tool provider by EMVCo, on successful completion of the L3 qualification process.
L3 QSP	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.
L3TG	Level 3 Testing Group. The new EMVCo-approved name assigned to the Terminal Integration Task Force (TITF).
M	
Mobile Point-of-Sale/Mobile Point-of-Service (MPOS or mPOS) device	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
M-TIP	Mastercard Terminal Integration Process. The Mastercard terminal integration testing process.
O	

Offline-capable	A transaction acceptance device that has the ability to process transactions offline, making use of the EMV offline authorization functionality.
Offline-only	A transaction acceptance device that is only able to process transactions offline. For example, low value POS or road toll terminals.
Online-capable	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.
P	
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.
Participant System	An entity (e.g., domestic payment systems, global payment systems and other similar entities) that has been assigned a L3 Participant System Identifier (PSI) through EMV's registration service to enable the usage of the EMV L3 Testing Framework.
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 Participant 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 Participant 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 System Environment
PS	Participant System

PSI	Participant System Identifier
R	
RFU	Reserved For Future Use
RID	Registered Application Provider Identifier
S	
SDA	Static Data Authentication
T	
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 Participant Systems, aimed at providing a level of assurance that Participant System-specific requirements and recommendations are being implemented in contact or contactless chip acceptance device designed specifically to interoperate with the Participant Systems' networks.
Terminal Integration Task Force (TITF)	Task Force established by EMVCo in 2013 with the purpose of examining each of the Participant 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 Participant 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 Participant System requirements.
Test Plan	A Participant 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 Participant 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 Participant 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 *not* acting as a merchant, acquirer, issuer or Participant 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
U	
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.
V	
var.	Variable
VAR	Value-added Reseller
X	
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

Annex B Tool Pass/Fail Automation Criteria – DataItem

Introduction

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.

The following tables define the format used for the following Data Items: NET, APDU, ATR and MANUAL. The separate levels have been noted as follows: Level 1 (L1) for the top level, Level 2 (L2) to designate a lower level, etc.

General requirements applicable throughout the following tables:

- If the Data Item cannot be found in the log when available, then the check result shall be considered as false (except for a check using the "not exists" operator).
- 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.
- The .BYTE syntax may be further qualified to indicate a particular bit using the syntax .BIT.OFFSET-LEN. The -LEN parameter starts at 1. However, the -LEN parameter is only present if the value is not 1. The .OFFSET and -LEN notation shall be in decimal values.
    - **Warning:** 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.
- 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 starts at 1. However, the -LEN parameter is only present if the value is not 1. The .OFFSET and -LEN notation shall be in decimal values.
    - **Warning:** Care needs to be taken since the value specified is a byte offset and not a byte number. Zero represents the first byte found in the data when it is considered to be a byte string. For example, the byte string "FF00000000" is set to FF at offset 0.
- 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. 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.

For presence data, the following notation is used:

- M: Mandatory – shall always be present.
- C: Conditional – shall be present unless the condition is not met.
- O: Optional – may or may not be present.

For occurrence data, the following notation is used:

- 1..1: Element is mandatory and can only occur once.
- 0..1: Element is optional and if present, can only occur once.
- 1..n: Element is mandatory and can occur recursively more than once.
- 0..n: Element is optional and can occur recursively more than once.

DataItem Format Requirements

Table B.1: NET DataItem Format

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	M/C/O	Occurrence	Description and Requirement
NET												M	1..1	The data to be checked is from a Network message. 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. 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.
.	XXXX											M	1..1	The network message identifier.
	IN or IL											O	0..1	Optional indicator that the data item represents instance N of the message. IL could also be used to represent the last instance of the message.
	.	BYTE										O	0..1	
		.	OFFSET-LEN									O	0..1	
	.	TAG										O	0..n	If the NET is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation.
		.	TAG_NO									C	0..1	Mandatory if TAG is present
			.	BYTE								O	0..1	
				.	OFFSET-LEN							O	0..1	
	.	DE										O	0..1	Indicates which Data Element (DE) NNN in the network message should be checked.
		.	NNN									C	0..1	Mandatory if DE is present.
			.	BYTE								O	0..1	
				.	OFFSET-LEN							O	0..1	

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	M/C/O	Occurrence	Description and Requirement
			.	TAG								O	0..n	If the DE is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation.
				.	TAG_NO							C	0..1	Mandatory if TAG is present
					.	BYTE						O	0..1	
						.	OFFSET-LEN					O	0..1	
			.	SE								O	0..n	In the case where a data element is made up of a number of Sub Elements (SE), the SE notation may be used to indicate which SE should be checked.
				.	NNN							C	0..1	Mandatory if SE is present.
					.	BYTE						O	0..1	
						.	OFFSET-LEN					O	0..1	
					.	TAG						O	0..n	If the SE is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation.
						.	TAG_NO					C	0..1	Mandatory if TAG is present.
							.	BYTE				O	0..1	
								.	OFFSET-LEN			O	0..1	
					.	SF						O	0..n	Indicates that Sub-Field NNN should be checked.
						.	NNN					C	0..1	Mandatory if SF is present.
							.	BYTE				O	0..1	
								.	OFFSET-LEN			O	0..1	
							.	TAG				O	0..n	If the SF is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation.
								.	TAG_NO			C	0..1	Mandatory if TAG is present.
									.	BYTE		O	0..1	

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	M/C/O	Occurrence	Description and Requirement
										.	OFFSET-LEN	O	0..1	

Examples:

NET.0100 = The authorization request in the network transaction log.

NET.0100I2 = The second authorization request in the network transaction log from the start of this test case.

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.

NET.0100.DE.055.TAG.9F36 = The ATC in an authorization request.

NET.0100.DE.055.TAG.95.BYTE.0-4 = The first four bytes of the TVR in an authorization request.

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).

NET.????.DE.048.SE.022.SF.001 = Sub Field 001 of sub element 022.

NET.0100.DE.055.TAG.95.BYTE.0.BIT.0 = TVR: No ODA performed

NET.0100.DE.055.TAG.95.BYTE.0.BIT.1 = TVR: SDA failed

NET.????.DE.048.SE.022.SF.001.SF.004 = Sub Field 004 of sub Field 001

NET.0100.DE.054.SE.004.BYTE.1-6 = 6 bytes starting from the 2nd byte of sub element 004 within data element data element 054.

NET.0100.DE.104.TAG.5C.TAG.1F11 = Value of Tag 1F11 (Primitive Data Object) in Tag 5C (Constructed Data Object) in data element 104.

Note: 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 Participant System's specific mapping tables are described in the following documents:

Table B.2: Non-ISO PS Mapping Document Reference

Participant System's name	Mapping document reference

Table B.3: APDU DataItem Format

L1	L2	L3	L4	L5	L6	L7	M/C/O	Occurrence	Description
APDU							M	1..1	The data to be checked is from an APDU that has been exchanged with the card. The optional P1, P2, LC, DA & LE values shall be present as defined in the EMV command identified by CCII. If any optional value (except IN/IL) is present then the previous optional values in the list shall also be present. It is not necessary for all the optional values to be present. 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 until the first match unless using IN or IL. 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.
.	CC						M	1..1	Hex CLA byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	II						M	1..1	Hex INS byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	P1						O	0..1	Optional P1 byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	P2						O	0..1	Optional P2 byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	LC						O	0..1	Optional LC byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	

L1	L2	L3	L4	L5	L6	L7	M/C/O	Occurrence	Description
	DA						O	0..1	Optional Data of length LC
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	LE						O	0..1	Optional LE byte
	.	BIT					O	0..1	
		.	OFFSET-LEN				O	0..1	
	IN or IL						O	0..1	Optional indicator that the data item represents instance N of the message. The last instance is represented by the keyword "IL".
	.	IFD					O	0..1	Represents data sent to the card by the interface device.
		.	BYTE				O	0..1	
			.	OFFSET-LEN			O	0..1	
		.	TAG				O	0..n	
			.	TAG_NO			C	0..1	Mandatory if TAG is present
				.	BYTE		O	0..1	
					.	OFFSET-LEN	O	0..1	
	.	ICC					O	0..1	Represents data returned by the card in response to the command.
		.	BYTE				O	0..1	
			.	OFFSET-LEN			O	0..1	
		.	TAG				O	0..n	
			.	TAG_NO			C	0..1	Mandatory if TAG is present
				.	BYTE		O	0..1	
					.	OFFSET-LEN	O	0..1	
	.	SW12					O	0..1	Represents the status words returned in the APDU response.
		.	BYTE				O	0..1	
			.	OFFSET-LEN			O	0..1	

Examples:

APDU.80AE = Generate Application Cryptogram Command.

L1	L2	L3	L4	L5	L6	L7	M/C/O	Occurrence	Description
APDU.80AEI2 = Second Generate Application Cryptogram Command. APDU.00B2010C = Read Record 1 File 1 Command. APDU.00B2 = Any read record command (since P1 is not specified). APDU.00A4040008A000000003101001 = The Select command which sends the data A000000003101001 (LE value is not present). APDU.00B201.ICC.TAG.70.TAG.61.TAG.50 = Application Label read from record 1. APDU.80A8.IFD.BYTE.2-2 = First two bytes of PDOL data sent in GPO APDU. APDU.80A8.SW12 = Status words returned from GPO command. APDU.80AE.IFD.BYTE.0-6 = GEN AC Amount value sent (assuming that Amount, Authorized is the first data object in CDOL1). APDU.80AE.P1.BIT.3 = Bit at offset 3 in the GEN_AC P1 parameter. APDU.80AE.ICC.TAG.77.TAG.9F10 = Issuer Application data in GEN AC response. APDU.80AE.ICC.TAG.77.TAG.9F10.0-2 = the 2 first (left most) bytes of the Issuer Application Data APDU.80AE.ICC.TAG.77.TAG.9F10.3 = the fourth byte of the Issuer Application Data APDU.80AE.ICC.TAG.77.TAG.9F10.3.BIT.0 = the left most bit of the fourth byte of the Issuer Application Data APDU.00A4040008A00000000310100100.ICC.TAG.87 = the Application Priority Indicator returned from the Select command which sends the data A000000003101001. APDU.00A40402??A000000003.SW12 = Check the Status Words returned for the “next occurrence” Select command which includes data A000000003.									

Table B.5: Manual DataItem Format

L1	M/C/O	Occurrence	Description
MANUAL	M	1..1	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.

Other usage of the data item field

The data item field may also be in the form LENGTH (<data_item>) 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.

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.

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:

- TT_ARQC_VALIDATE – the L3TT shall validate the ARQC as defined in the emvcard.ac() pseudo-function. The L3TT shall use the corresponding L3-OML to validate the cryptogram.
- SIM_ARQC_VALIDATE - Check that the simulator has validated the ARQC correctly
- SIM_PIN_VALIDATION - Check that the simulator has validated the PIN
- SIM_TRACK1CVC3_VALIDATE - Check that the simulator has correctly validated the Track1 CVC3 value
- SIM_TRACK2CVC3_VALIDATE - Check that the simulator has correctly validated the Track2 CVC3 value
- LOG_RESET - Reset log position to the beginning of the log for this test case

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