EMV® Specification No. 251
First Edition, December 2020

EMVCo Level 3 Framework Implementation Guidelines Updates

This EMVCo Level 3 Testing Group (L3TG) Technical Bulletin provides details of updates, clarifications and corrections required to the L3 EMV® Level 3 Testing Framework Implementation Guidelines – Version 1.1 (dated April 2019).

Applicability

This Bulletin applies primarily to:

- *EMVCo Level 3 Solutions and Service Providers, including (but not limited to):*
    - *Suppliers of Level 3 test tools*
    - *Entities performing L3 test tool qualification testing on behalf of EMVCo*

Related Documents

- *EMV Level 3 (L3) Testing Framework Implementation Guidelines – Version 1.1 (dated April 2019)*

Dates

- 1st edition of the Bulletin effective date: December 9th, 2020
- EMVCo qualification of the L3 Test Selection Engine (L3TSE) component, in accordance with FIG v1.1 + bulletin 251 starting date: December 14th, 2020
- EMVCo qualification of the L3 Test Tool Engine (L3TT) component, in accordance with FIG v1.1 + bulletin 251 starting date: December 14th, 2020
- EMVCo qualification testing of FIG v1.1.242 and below will no longer be supported as of December 14th, 2020

Description

Since publication of the *L3 Testing Framework Implementation Guidelines – Version 1.1* document in April 2019, feedback has been received by EMVCo on some areas that require further clarification or correction.

This L3TG Technical Bulletin provides updates, clarifications and corrections required.

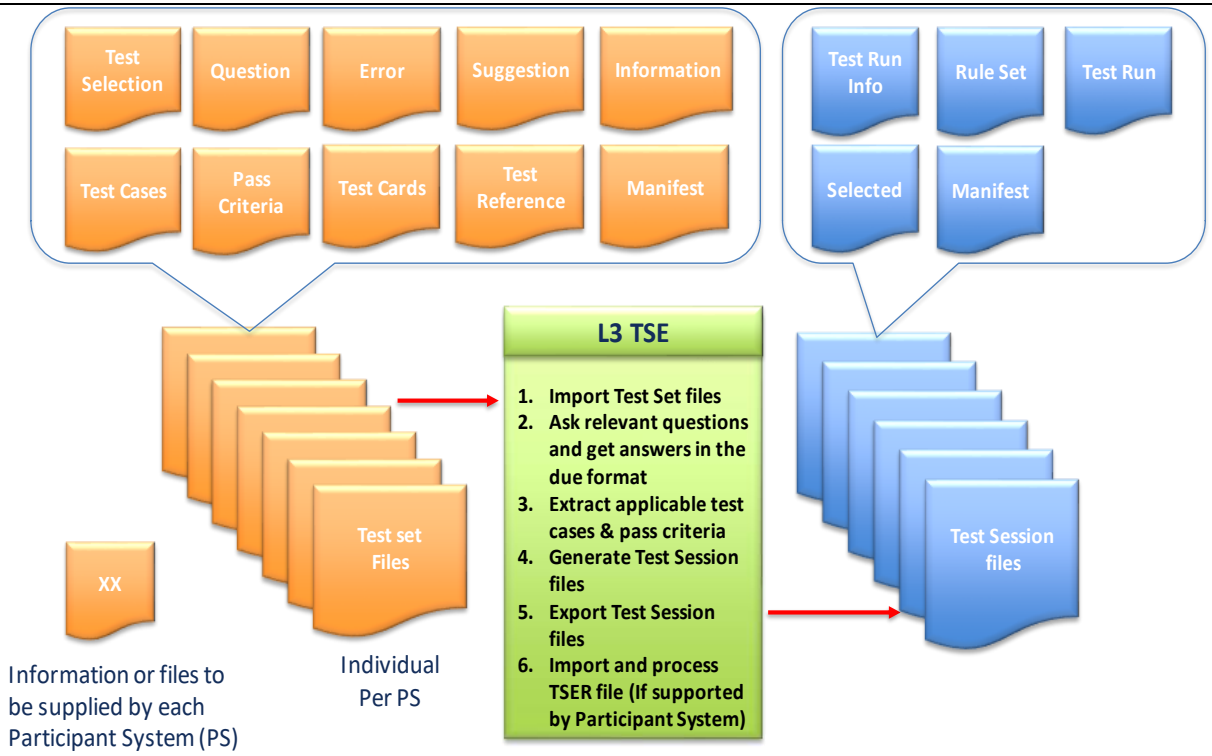
Impacted components: L3TT and L3TSE.

Document Updates:

This section documents the updates required to the *L3 Testing Framework Implementation Guidelines – Version 1.1* document. Correction instructions are shown in blue, while the updates required within existing texts are shown in red (L3TB242) and in green.

Important note: for test tool vendors electing to support the L3TSE component, the common question feature is a mandatory requirement which must be implemented (refer to req # 6.1 and section 4.1.3 for details).

Item #	Section:	Update required:
1.1	2 L3 Test Environment Architecture	In L3 TSE component, update as follows: **L3 TSE:** Third-party vendor-provided Test Selection Engine is intended to provide Clients (that often support multiple Payment Systems) with a convenient means of preparing applicable Test Session files for individual Payment Systems. The Test Session files are subsequently presented in a machine-readable format that will be compatible with any EMVCo-qualified L3 Test Tool. EMVCo will qualify the L3 TSE's capability to: • Import the machine-readable TSE Test Set files provided as a TSEC-package by the Payment Systems. There will be one or more TSEC-packages per Payment System. It is a zip file with Test Set files which include instructions on how to collect terminal configuration information, test cases, test case applicability conditions, and pass criteria definitions. This can happen for multiple Payment Systems, • Process the configuration provided by the TSE Test Set files by collecting the answers to the applicable set of questions presented to the user. Answering these questions is required in order to evaluate the L3 testing scope. If the client's terminal supports multiple Payment Systems, then this process needs to happen for each applicable TSEC-package separately. • Extract the applicable test cases and related pass criteria. • Build the corresponding individual TSE Test Session files that describe the applicable test plan for the terminal under test. • Export the individual Test Session files grouped in a TSE-package (one per TSEC-package). This is a zipped folder with renamed extension .tse ready to be used by a L3TT tool. • 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.
1.2		In Figure 2-1, update as follows:

		
2	3.1 Definitions	In the **Context expression** section, update as follows: 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 should be considered as not applicable. * Otherwise: it should 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. **Timestamp** Always in ISO-8601 format unless specified differently. [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*.

<table>
<tr><td></td><td></td><td>After the Timestamp term definition, update as follows:
Project
A project is defined as the testing of one particular Terminal Integration solution deployed across one or several PSI(s).</td></tr>
<tr><td>3.1</td><td rowspan="2">3.2 TSE Requirements</td><td>In Requirement 2.4, update as follows:
• The manifest file shall have the correct signature (if present).</td></tr>
<tr><td>3.2</td><td>New/update Requirement:
Add/update the following to the existing Requirements table:
<table>
<tr><th>Req #</th><th>Requirement</th><th>M/O/C</th></tr>
<tr><td>5.0</td><td>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 <br>).
It is not acceptable when the HTML codes (like) 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.</td><td>M</td></tr>
<tr><td>6.1</td><td>• L3TSE shall evaluate the Question applicability for each of the questions to understand whether the question shall be presented to the user.</td><td>M</td></tr>
<tr><td>6.1.1</td><td>– The answer to a previously presented question with attribute “Common” shall, be treated as a suggestion within the same project. It should be handled as a “suggest” SuggestionType. Suggestions originating</td><td>M</td></tr>
</table></td></tr>
</table>

			from common answers shall be treated after regular suggestions.	
		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
		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
		15.2	L3TSE shall ask the user whether or not he wants to upgrade to the latest Build. If not, then the procedure as described in requirement 14 applies, if yes, then continue as described below.	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
		18.4	• L3TSE shall update the original .tse file with the info provided in the validation report (.tser), if confirmed by the user.	M

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>18.5</td><td>• 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: Passed or Failed
− 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</td><td>M</td></tr>
<tr><td>18.6</td><td>• 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.</td><td>M</td></tr>
<tr><td>18.7</td><td>• 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.</td><td>M</td></tr>
<tr><td>18.8</td><td>• The Validation Report .tser file shall be added to the L3TSE session as a global attachment.</td><td>M</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>4. EMVCo L3 Detailed Formats</td><td>This section describes the details of the different formats of each component.
Note: as a general requirement, the file extension and file name shall not be case sensitive. In addition, as EMVCo reserves the right to add or remove data in the xmls, test tools shall be able to ignore the addition or removal of data. This is in order to ensure backwards and forward compatibility between versions.</td></tr>
<tr><td>5.1</td><td>4.1.1 Test Set File Considerations</td><td>In the Character Substitutions section:
In first paragraph, updates as follows:
The characters used to structure fields may need to be part of the data itself. In order to allow these character codes a number of substitutions shall take place when the output file is created. These substitutions shall be reversed by tools processing the TSE Test Set files. Characters (listed in table 4.1) defined as part of the format of the field shall not be substituted.
[…]</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Note: Some HTML codes include character substitution as describe above, the substitutions shall be reversed before HTML tags are interpreted.</td></tr>
<tr><td>5.2</td><td></td><td>In second paragraph, update as follows:
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.</td></tr>
<tr><td>5.3</td><td></td><td>In the Bit Definition Fields section:
Make the following updates to the table:
<table>
<tr><th>Context expression</th><th>Bitmap position</th><th>Context Statement</th><th></th></tr>
<tr><td>Interface=Contact</td><td>Bit 1</td><td>True</td><td>Least significant</td></tr>
<tr><td>Interface=Contactless</td><td>Bit 2</td><td>True</td><td></td></tr>
<tr><td>Term_type=Attended POS</td><td>Bit 3</td><td>True</td><td></td></tr>
<tr><td>Term_type=ATM</td><td>Bit 4</td><td>False</td><td>Most significant</td></tr>
</table>
The evaluation of context expressions shall not be case sensitive, e.g. “Term_type=Attended POS” is the same as “term_Type=attended pos.</td></tr>
<tr><td>5.4</td><td></td><td>Add the following new table between the existing table and the text starting “As the Terminal Integration context…”:
<table>
<tr><th>Bitmap position</th><th>Bit 4 - most significant bit</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1 - least significant bit</th></tr>
<tr><td>Context expression</td><td>Term_type=ATM</td><td>Term_type=Attended POS</td><td>Interface=Contactless</td><td>Interface=Contact</td></tr>
<tr><td>Context Statement</td><td>False</td><td>True</td><td>True</td><td>True</td></tr>
</table></td></tr>
<tr><td>5.5</td><td></td><td>In third paragraph, update as follows:
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).</td></tr>
<tr><td>5.6</td><td></td><td>Add a new paragraph under the third paragraph as follows:</td></tr>
</table>

<table>
<tr><td></td><td></td><td>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).</td></tr>
<tr><td>5.7</td><td></td><td>In the Applicability of Context-related Data section, add a note under the last paragraph as follows:
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.</td></tr>
<tr><td>5.8</td><td></td><td>In the TSE Test Set Files Format section, update as follows:

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*””)”).</td></tr>
<tr><td>5.9</td><td></td><td>In the Naming Conventions section, updates as follows:
TSE Test Set files are versioned using
• P: EMVCo L3 Participant System Identifier,
<table>
<tr><th>Identifier</th><th>Participant</th></tr>
<tr><td>00</td><td>EMVCo</td></tr>
<tr><td>01</td><td>American Express</td></tr>
<tr><td>02</td><td>Discover</td></tr>
<tr><td>03</td><td>JCB</td></tr>
<tr><td>04</td><td>Mastercard</td></tr>
<tr><td>05</td><td>UnionPay</td></tr>
</table>
</td></tr>
</table>

		<table><tr><td>**06**</td><td>Visa</td></tr><tr><td>**07 … 99**</td><td>Reserved for future Participants</td></tr></table> • N: Participant System's Test Set file 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 Participants shall ensure that a series or a build is generated following a logical naming 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).
6	4.1.11 Manifest File Format	Add a new paragraph before File Format as follows: **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 PSI 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 bulletin would be: 1.1.251 additional examples: 1.2, 1.3, 1.3.001, 1.3.002, 2.0, etc). In the table, update as follows: PSI =00 Series =0 Build =140

		Name =DemoPlan Mode =EMVCoL3 Version =1.1.251 Selection_File =00_DemoPlan_0_140_selection.csv: LEy6saV522S2l8h+jwPxcKYW7I8=: Scenario_File =00_DemoPlan_0_140_test_case.csv: MXbW1fj+qZYAUa1fvJbOohVjGxs=: Question_File =00_DemoPlan_0_140_question.csv: j13wfAVBV3rr/p+0A3ZVTSqWNC0=: Error_File =00_DemoPlan_0_140_error.csv: V9bh7yrF8s0KMgStFJ0kbmenxbo=: Suggest_File =00_DemoPlan_0_140_suggestion.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_information.csv: KmfR/M5S6TCAc2svwvZ4t963NiM=: Test_Reference =00_DemoPlan_0_140_test_reference.html: eI4Pj5tA0EHL8XHYOtbbseLe7qg=: Signature =vzFajPLsUNk= 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 ':'. **Note:** a TSEC-package may optionally include additional files (e.g. release notes.txt) 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. The manifest file shall not include any duplicate lines.
7	4.1.2 Test Selection File Format	In the **Context Expression 1..n** Data Field of the table, update as follows: One entry per context expression is used to define the Terminal integration context. There is a dedicated field for each bit 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"
8.1	4.1.3 Question Definition File Format	In the **Context Questions section**, update as follows: **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.
8.2		In the **Type** data field of the table in the **Common Questions** section, update as follows: 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 section 6.6.

9	4.1.5 Suggestion Definition File Format	Add the following rule in "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": 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 should 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, 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
10.1	4.1.8 Test Case File Format	In the **Name** data field of the table, update as follows: Unique name that corresponds to a test case name in the Test Selection file
10.2		In the **Cards** data field of the table, update as follows: 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.
11.1	4.1.11 Manifest File Format	Add the following after the last paragraph: Mode: Main usage of the Test Set file. Predefined value is “EMVCoL3” Integrity: The SHA-1 hash of the file converted to a Base 64 string Signature (Optional): 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.
11.2		In the **File Format** section, first line of the example, update as follows: • PSI = 00
12.1	4.2 TSE Test Session Files	In the last bullet point of “The TSE Test Session files are the following XML files archived in a .tse file (a renamed .zip file):”, update as follows: • Manifest.txt – Mandatory - Provides a protected list of all the files output and binds them together into a single data set
12.2		In the “typical header & trailer example”, update as follows:

<table>
<tr><td></td><td></td><td>

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

<!--pEkDJSlje98OIPki5w/fDQXWKjE=-->
```

</td></tr>
<tr><td>13.1</td><td>4.2.1
TestRunInfo.xml</td><td>In the TestRunInfo.xml table, add the following tags to the table (after ….<SelectedTests>):
<table>
<tr><th>Section – Block tag – Data field tag</th><th>Description</th><th>L3TSE Requirement</th></tr>
<tr><td>….<L3TSEFIGVersion></td><td>Version of the L3 Test Selection Engine (including potential bulletin) used at the time of the test run (e.g. 1.1.242, 2.0, etc).</td><td>Required</td></tr>
<tr><td>Section – Block tag – Data field tag</td><td>Description</td><td>L3TT Requirement</td></tr>
<tr><td>….<L3TTFIGVersion></td><td>Version of the L3 Test Tool Engine (including potential bulletin) used at the time of the test run (e.g. 1.1.242, 2.0, etc).</td><td>Conditional
Condition: if a L3TT is used.</td></tr>
</table>
</td></tr>
<tr><td>13.2</td><td></td><td>In the <Mode> data field, make the following updates to its Description:
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 is being displayed
MODE_TestResult: All questions have been answered and test results are being recorded</td></tr>
</table>

		MODE_Regression Questions have been changed and the user should be prompted to clear regression test results. <Mode> flow: • User opens TSE and starts answering questions -> MODE_Question • All questions are answered and the test information is being displayed -> MODE_Info • Test results are being recorded -> MODE_TestResult • User goes back to the questions and changes the answer to a question with attribute=regression -> MODE_Question • All questions are answered and the test information is being displayed -> MODE_Info • User confirms the information report -> MODE_Regression • User is prompted to clear regression test results (test cases marked as "isRegression=Yes") and confirms -> MODE_TestResult
14.1	4.2.2 RuleSet.xml	In the **Error Definition File** field of the table, update as follows: <Errors> ..<Error> ....<mask1> ....<mask2> ....<name> ....<ErrorType> ....<message> ....<GotoQuestion>
14.2		In the **Question Definition File** field of the table, update as follows: ....<allowed> ….<attributes>

<table>
<tr><td></td><td></td><td>....<prompt>
....<help>
....<mode>
....<groups></td></tr>
<tr><td>14.3</td><td></td><td>In the Substitutions section, update as follows:
As described in the Test Set files definition, the CSV data contains various substitutions so that characters can be used to delimit structured fields. See Table 4.1. The substitutions shall not be passed on to the XML file, e.g. “ &ob” shall be replaced by “[“ in the XML. The following separators: &cm, &sc and &cl shall be kept as such in the XML, to avoid ambiguity. XML has its own set up substitution rules that shall be applied (regardless of the format of the field):</td></tr>
<tr><td>15.1</td><td>4.2.3 TestRun.xml</td><td>In the TestRun.xml section Update table, update as follows:

<table><tr><td>….<Answer></td><td>The answer as given during the L3 TSE process to this question

The multiple answers (e.g. from VarSet type question) shall be coded as per the example below:
<Response>
<Name>Input_VarSet</Name>
<Answer>[def][ghi]</Answer>
<Updated>2020-06-26T 07:00:10:609Z</Updated>
</Response></td></tr></table></td></tr>
<tr><td>15.2</td><td></td><td>In the <Files> sections of the “Update” table, make the following updates:
Comma separated of quoted Log file name(s) + optional directory path e.g.</td></tr>
</table>

		“c:\Logfiles\Logfile1.xml","c:\Logfiles\Logfile2.xml”,“Logfile3.xml” Comma separated of quoted Log file name(s) + optional directory path e.g. “c:\Logfiles\Logfile4.xml","c:\Logfiles\Logfile5.xml”,“Logfile6.xml” **Note: <Files>** may include additional log file formats.
15.3		At the end of section 4.2.3, add the following note: **Note:** If a check step is not executed, then the <Result> shall be left to “not tested”.
16.1	4.3 Tool Pass/Fail Automation Criteria	In Table 4.3, in the description of **DataItem** tag, update as follows: 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. If the Data Item cannot be found then the check result shall be considered as false (except for a check using the “not exist” operator). The name is a hierarchical description of the data item using a "dot" notation to separate levels. The intention is that the description uniquely and unambiguously describes what data need to be checked using a name that can be understood by automated tools. At the top level the notation starts with: NET.XXXIN - The data to be checked is from a Network message − XXX = the network message identifier. The ? character may be used as a wildcard to indicate any valid character. For example, "NET??10" indicates any network message that has an identifier ending in 10. 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. − IN = optional indicator that the data item represents instance N of the message

		APDU.CCIIP1P2LCDALEIN - The data to be checked is from an APDU that has been exchanged with the card − CC =Hex CLA byte − II = Hex INS byte − P1 = Optional P1 byte − P2 = Optional P2 byte − LC = Optional LC byte − DA = Optional Data of length LC − LE = Optional LE byte − IN = optional indicator that the data item represents instance N of the message. The last instance is represented by the keyword “IL”. 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. 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. ATR.MODE - The data to be checked is from the "Answer To Reset" from the card − MODE = "COLD" - indicates the ATR from the card cold reset − MODE = "WARM” - indicates the ATR from the card warm reset • MANUAL - This is a manual check that can't be completed automatically. The value column (see below) gives a textual description of the check to be performed Some example uses of the top level notation would be: 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 APDU.80AE = Generate Application Cryptogram Command APDU.80AEI2 = Second Generate Application Cryptogram Command APDU.00B201 = Read Record 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) ATR.WARM = Bytes from the warm Answer to Reset The top level notation is further qualified by additional "dot" elements that further qualify the data item. In the case of NET data items the further qualification is as follows: − NET − DE.NNN - indicates data element NNN should be checked ○ SE.NNN - indicates that sub-element NNN should be checked ▪ BYTE.OFFSET-LEN ▪ TAG.TAG_NO − SF.NNN – indicates that sub-field NNN should be checked − BYTE.OFFSET-LEN − TAG.TAG_NO ○ BYTE.OFFSET-LEN ○ TAG.TAG_NO An additional DE qualification is added to indicate which data element in the network message should be checked. Individual bytes of this data element may be checked by using the .BYTE notation, alternatively if the data element is formatted as a Tag Length Value (TLV) structure then the tag of interest may be indicated using the .TAG notation. 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. In the case where a data element is made up of a number of sub elements the SE notation may be used to indicate which sub element should be checked. Some typical examples would be: 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

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.

Some examples are:
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

Warnings:

- 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.
- 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 .TAG syntax may include further child .TAG qualifications in the case where a nested TAG structure is present in the data to be checked.

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 payment system’s specific mapping tables are described in the following documents:

Payment System’s name	Mapping document reference

In the case of APDU data elements the further qualification is as follows:

		- APDU
		- P1 - Represents the P1 data sent to the card by the interface device
		  - BIT.OFFSET-LEN
		- P2 - Represents the P2 data sent to the card by the interface device
		  - BIT.OFFSET-LEN
		- *LC - Represents the number of data sent to the card by the interface device*
		  - *BIT.OFFSET-LEN*
		- *DA - Represents data sent to the card by the interface device or data returned from the card to the interface device*
		  - *BYTE.OFFSET-LEN*
		- *LE - Represents the expected number of data returned by the card in response to the command*
		  - *BIT.OFFSET-LEN*
		- IFD - Represents data sent to the card by the interface device
		  - BYTE.OFFSET-LEN
		  - TAG.TAG_NO.BYTE
		- ICC - Represents data returned by the card in response to the command
		  - BYTE.OFFSET-LEN
		  - TAG.TAG-NO
		- SW12 - Represents the status words returned in the APDU response
		  - BYTE.OFFSET-LEN
		
		Some examples are:
		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.80AEI1.IFD.TAG.95.BYTE.4 = the right most byte of the TVR
		APDU.80AEI1.IFD.TAG.95.BYTE.0-2 = the 2 first (left most) bytes of the TVR
		APDU.80AEI1.IFD.TAG.95.BYTE.0.BIT.0 = the left most bit of Byte 1
		APDU.00A4040008A00000000310100100.ICC.TAG.87 = the Application Priority Indicator returned from the Select command which sends the data A000000003101001

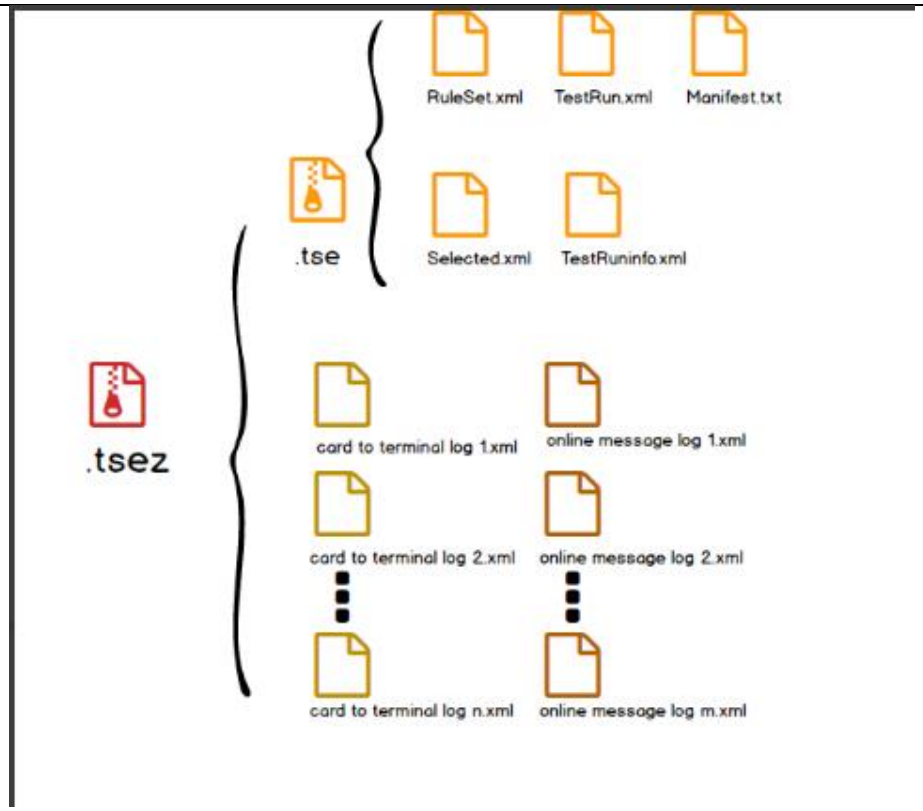
		APDU.00A40402??A000000003.SW12 = Check the Status Words returned for the “next occurrence” Select command which includes data A000000003 In the case of ATR elements the only possible further qualification is .BYTE.OFFSET-LEN to indicate a particular byte of the response: ATR.COLD.BYTE.0-5 = The first 5 bytes of the ATR 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 depend 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: − SIM_ARQC_VALIDATE - Check that the simulator has validated the ARQC correctly − SIM_PIN_DECRYPTION - Check that the simulator has correctly decrypted the PIN − 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 (see below) 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.
16.2		In Table 4.3, under **DataFormat**, add the following as the last paragraph: 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'.
16.3		In Table 4.3 under **Operator**, update as follows: 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
16.4		In Table 4.3 under **Value**, update as follows: 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 **Note 1:** Except for the BINARY function, the value used in the functions above shall not perform any conversion.

		Note 2: 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 (see section 4.3). • 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
16.5		In Table 4.3 under **ActionIfTrue**, update as follows: 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.
16.6		In Table 4.3 under **ActionIfFalse**, update as follows:

		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.
17	4.4 Test Report	Make the following updates to this section: The Test Report file is an archive file renamed .tsez, structured as shown below and made of: • The .tse file as described in chapter **Error! Reference source not found.**, • The test session logs files made of Card Terminal logs and On-line messages respectively described in chapter **Error! Reference source not found.** and **Error! Reference source not found.**. The test session logs are all located at the archive root.

		 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).
18	4.5.2 EMVCo Level 3 Card Image Definition	Update the list as follows: 2. Header information is included, such as:

<table>
<tr><td></td><td></td><td>[…]
f. L3FIG version containing the version of the FIG used to generate the Card Image Definition.xml (conditional if supported by the participant system).
g. L3PF version containing the version of the EMV L3 Pseudo Functions used to generate the Card_Image_Definition.xml (conditional if supported by the participant system).</td></tr>
<tr><td>19</td><td>4.6.3 Log Details</td><td>Add to the list as follows:
5. <L3FIGVersion> - This field contains the version of the FIG used by the L3CS component at the time of the test run (mandatory).
6. <L3PFVersion> - This field contains the version of the EMV L3 Pseudo Functions used by the L3CS component at the time of the test run (mandatory).
7. <CardId> - This field contains the name of the card present in the <CardId> tag of each Card_Image_Definition.xml at the time of the test run (conditional if present in the Card_Image_Definition.xml).
8. <CardVersion> - This field contains the version of the card version present in the <CardVersion> tag of each Card_Image_Definition.xml at the time of the test run (conditional if present in the Card_Image_Definition.xml).
[…]
Update Table 4.6 as follows:
<?xml version="1.0" encoding="utf-8"?>
<CardTerminalLog>
…
<LogDetails>
<i>Date-Time</i>>2013-08-21T10:43:46Z</<i>Date-Time</i>>
<<i>LoggingTool</i>>
<<i>ProductName</i>>Test Tool Name</<i>ProductName</i>>
<<i>ProductVersion</i>>1.0</<i>ProductVersion</i>>
<L3FIGVersion>1.1.242</L3FIGVersion>
<L3PFVersion>1.4</L3PFVersion>
</<i>LoggingTool</i>>
<<i>SchemaSelectionIndex</i>>1</<i>SchemaSelectionIndex</i>></td></tr>
</table>

```
                <Reference>SE:502101143;Test Case: XYZ-POS 001</Reference>
                <CardId>MP50</CardId>
                <CardVersion>1.0</CardVersion>
            </LogDetails>
            …
</CardTerminalLog>
```


20.1 — 4.7.2 Format Specification

In the third bullet, update as follows:

- 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.

20.2

In Table 4.14, update as follows:

Field	Block/Tag/Attribute	M/O/C	Description
In **<Protocol>** SymbolicName	Tag	M	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
[…]			
In **<Connection>** (after **TCPIPParameters**, block) <DialupParameters>	Block-start	C	Block which specifies the details of a PSTN connection in sufficient detail that a knowledgeable user is able to recreate the connection.

		PhoneNumber	Tag	M	For a client connection, this tag specifies the telephone number of the remote server which this connection must dial in order to initiate a call. For a server connection, this tag can remain blank.	
		ModemInitString	Tag	M	The modem initialization string delivered to the modem when the application starts and which is intended to set the modem to a known functional state sufficient for a call to be initiated.	
		Client	Tag	M	Specifies whether the connection is operating as a client (true) or a server (false).	
		Format	Tag	M	A free-format text string which describes the basic format of the data, eg, “ASCII”, “EBCDIC”, “BCD” etc.	
		</DialupParameters>	Block-end	C		
		[…]				

		In **<LogDetails>** (after **SchemaSelectionIndex** field) L3OMLVersion	Tag	M	This tag contains the version of the FIG used to generate the OML at the time of the test run.	
		PINValue	Tag	O	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.	
		<FieldList>	Block-start	M	Contains Field objects	
		ID	Attribute of tag *<Field>*	M	Tag 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 software to identify the field uniquely and is not user-visible. The ID value shall be coded as the Network Message-related "DataItem" value defined in the Check definition (Refer to Chapter	

<table>
<tr><td></td><td></td><td><table>
<tr><td></td><td></td><td></td><td>Error! Reference source not found. Tool Pass/Fail Automation Criteria). E.g. NET. ????.DE.048.SE.022.SF.001</td></tr>
<tr><td>PrefixBinary</td><td>Tag</td><td>C</td><td>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)</td></tr>
<tr><td>FieldBinary</td><td>Tag</td><td>M</td><td>Tag which specifies the raw, hexadecimal data for the field or subfield (example ‘F0 F1 F0 F0’).</td></tr>
<tr><td>FieldViewable</td><td>Tag</td><td>M</td><td>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 table 4.3 for details) without applying any conversion.</td></tr>
</table></td><td></td></tr>
<tr><td>21</td><td>4.8 TSE Validation Report Files</td><td colspan="3">Add the following as the first paragraph in this section:
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</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>

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: 4.8.1 - TSE Validation Report structure.
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.
</td>
</tr>
<tr>
<td>22</td>
<td>4.8.1 TSE Validation Report structure</td>
<td>
In Table in section 4.8.1, update as follows, the structure of the FIG was not aligned with XSD:

TSER file xml Format
<table>
<tr><th>Field</th><th>Block/Tag/Attribute</th><th>M/O/C</th><th>Description and Requirement</th></tr>
<tr><td><L3tse></td><td>Block-start</td><td>M</td><td>XML root node.</td></tr>
<tr><td>file_version</td><td>Attribute</td><td>M</td><td>Version of the format definition</td></tr>
<tr><td>originator</td><td>Attribute</td><td>M</td><td>Name of L3TSE tool generating the file</td></tr>
<tr><td><ValidationInfo></td><td>Block-start</td><td>M</td><td>L3TSE Validation Report file information</td></tr>
<tr><td><Context></td><td>Block-start</td><td>M</td><td>Single entry</td></tr>
<tr><td><TrackingNo></td><td>Tag</td><td>M</td><td>Tracking number of the .tse file that has been reviewed</td></tr>
<tr><td><OriginalFilename></td><td>Tag</td><td>M</td><td>Name of the .tse file that has been reviewed</td></tr>
<tr><td><RuleSetName></td><td>Tag</td><td>M</td><td rowspan="2">Record the version of CSV rules present in the .tse file at the time of review.</td></tr>
<tr><td><RuleSetSeries></td><td>Tag</td><td>M</td></tr>
</table>
</td>
</tr>
</table>

		<RuleSetBuild>	Tag	M	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.	
		<ReviewDate>	Tag	M	Records the data and time of the validation.	
		<ReviewStatus>	Tag	M	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.	
		<ReviewText>	Tag	O	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			
		</ValidationInfo>	Block-end			
		<Observations>	Block-start	M	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	Multiple entries	
		<TestName>	Tag	M		
		<Observation>	Tag	M		
		</Test>	Block-end			
		</Observations>	Block-end			

		<Answers>	Block-start	M	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	Multiple entries	
		<Name>	Tag	M		
		<Answer>	Tag	M		
		</Response >	Block-end			
		</Answers>	Block-end			
		<Results>	Block-start	M	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, L3SE 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	Multiple entries	
		<TestName>	Tag	M		
		<MajorVersion>	Tag	M		
		<CheckNo>	Tag	M		
		<StepNo>	Tag	M		

<table>
<tr><td></td><td></td><td>
<table>
<tr><td><Result></td><td>Tag</td><td>M</td><td>Shall be set to “pass” or “fail” or “not tested”</td></tr>
<tr><td></Test></td><td>Block-end</td><td></td><td></td></tr>
<tr><td></Results></td><td>Block-end</td><td></td><td></td></tr>
<tr><td></L3tse</td><td>Block-end</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>23</td><td>Annex A Common L3 Terminologies</td><td>Add the following in the table:
<table>
<tr><td>PSI</td><td>Participant Systems Identifier</td></tr>
</table>
</td></tr>
</table>

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 noninfringement. 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.

This document is provided "AS IS" without warranties of any kind, and EMVCo neither assumes nor accepts any liability for any errors or omissions contained in this document. EMVCO DISCLAIMS ALL REPRESENTATIONS AND WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AS TO THIS DOCUMENT.

EMVCo makes no representations or warranties with respect to intellectual property rights of any third parties in or in relation to this document. EMVCo undertakes no responsibility to determine whether any implementation of this document may violate, infringe, or otherwise exercise the patent, copyright, trademark, trade secret, know-how, or other intellectual property rights of third parties, and thus any person who implements any part of this document should consult an intellectual property attorney before any such implementation.

Without limiting the foregoing, this document may provide for the use of public key encryption and other technology, which may be the subject matter of patents in several countries. Any party seeking to implement this document is solely responsible for determining whether its activities require a license to any such technology, including for patents on public key encryption technology. EMVCo shall not be liable under any theory for any party's infringement of any intellectual property rights in connection with this document.