EPC Information Services (EPCIS) 1.1 Specification

Conformance Requirements Document

Version 1.21, December 2014

GS1 Global Office
Avenue Louise 326, bte 10
B-1050 Brussels, Belgium

Disclaimer

GS1 AISBL (GS1) is providing this document as a free service to interested industries.
This document was developed through a consensus process of interested parties in developing the Standard. Although efforts have been made to assure that the document is correct, reliable, and technically accurate, GS1 makes NO WARRANTY, EXPRESS OR IMPLIED, THAT THIS DOCUMENT IS CORRECT, WILL NOT REQUIRE MODIFICATION AS EXPERIENCE AND TECHNOLOGY DICTATE, OR WILL BE SUITABLE FOR ANY PURPOSE OR WORKABLE IN ANY APPLICATION, OR OTHERWISE. Use of this document is with the understanding that GS1 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF NON-INFRINGEMENT OF PATENTS OR COPYRIGHTS, MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE, THAT THE INFORMATION IS ERROR FREE, NOR SHALL GS1 BE LIABLE FOR DAMAGES OF ANY KIND, INCLUDING DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES, ARISING OUT OF USE OR THE INABILITY TO USE INFORMATION CONTAINED HEREIN OR FROM ERRORS CONTAINED HEREIN.

3 REVISION HISTORY:

Version	Date	Author	Modifications
1.0	12/15/06	Craig Asher, IBM	First draft of conformance requirements list for EPCIS SAG review
1.1	12/20/06	Craig Asher, IBM	Corrections made to conformance requirements list per input from Samsung and Polaris Networks
1.2	12/27/06	Craig Asher, IBM	Proposed triage of conformance test cases among the test case participants
1.3	1/4/07	Craig Asher, IBM	Added Notes on lines 95-110 to provide guidance to the writers of the Test Case Requirements based on our discussion in the EPCIS SAG meeting on 3 Jan 07
1.4	1/21/07	Hersh Bhargava, Rafcore; John Cooper, Kimberly-Clark; Richard Bach, GlobeRanger; Nagendra Revanur, T3Ci; Abel Sanchez, Auto-ID Labs; Uday Sadhukhan, Polaris Networks; Eliot Polk, Reva; Sunghak Song, Samsung; Craig Asher, IBM	Added conformance test case requirements; Craig checked & edited requirements
1.5	1/24/07	Nagendra Revanur, T3Ci; Abel Sanchez, Auto-ID Labs; Craig Asher, IBM	Added and corrected test cases
1.6	1/24/07	Scott Barvick, Reva Systems	Added structure to separate Capture App client operation from server so that specific Capture App-only certification is possible
1.7	1/28/07	Nagendra Revanur, T3Ci; Craig Asher, IBM	Added test cases T3Ci-10, 11, and 12; updated test case numbers in Section 2
1.8	1/29/07	Gena Morgan, EPCglobal	Modified Test Case identification keys
1.9	2/1/07	Gena Morgan, EPCglobal	Modified test cases per group review and con call on 1/31/07
1.10	2/7/07	Sunghak Song, Samsung; Craig Asher, IBM	Corrected test cases 2.2, 2.4, 2.5, 2.22, 2.39
1.11	2/7/07	Gena Morgan , EPCglobal	Replaced all references to original test case identifiers to new reference numbers. Corrected test cases per CA and Reva's email dated 2/7/07 #1,3, 4, 5 and 6. Still have questions on 2 and 7.
1.12	2/11/07	Craig Asher, IBM	Corrected test case 2.26, 2.39, 2.43 and 2.45 per request from Eliot Polk, Reva Systems
1.13	4/4/14	Hussam El-Leithy, GS1 US	Initial draft including changes made during EPCIS 1.1 development
1.14	4/29/14	Hussam El-Leithy, GS1 US	Incorporated feedback from Metlabs
1.15	5/6/14	Hussam El-Leithy, GS1 US	Updated line numbers to match final EPCIS 1.1 version
1.16	8/8/14	Hussam El-Leithy, GS1 US	Incorporated initial feedback from MetLabs

1.17	9/9/14	Hussam El-Leithy, GS1 US	Incorporated additional feedback from MetLabs
1.18	9/23/14	Hussam El-Leithy, GS1 US	Reinserted TCR#2 with rephrased Pre-test conditions and the step description
1.19	10/14/14	Hussam El-Leithy, GS1 US	Updated TCR#28 to be consistent with EPCIS 1.1
1.20	11/13/14	Hussam El-Leithy, GS1 US	Incorporated additional feedback received from community.
1.21	12/17/2014	Gena Morgan, GS1	Incorporated workgroup comments and changes

Abstract

This document outlines the approach to conformance testing for the GS1 EPCIS 1.1 Specification.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the GS1. This document has been reviewed by the working group and is in its final form of delivery to GS1.

Table of Contents

1 Functional Requirements

The EPCIS 1.1 Specification defines specific functionality that a valid EPCIS Implementation must provide. The following tables outline the specific requirements that must be tested as defined by the specification. Each test requirement entry references the EPCIS 1.1 Specification and the test case requirement (TCR) used to verify functionality as defined in section 2 of this document.

There are four main branches for our testing, where each branch has sub branches:

Test Branch 0 – Capturing Application

- 0A – Capturing Application using HTTP binding
- 0B – Capturing Application using message queue binding

Test Branch 1 – Capture Server

- 1A – Capture Server using HTTP binding
- 1B – Capture Server using message queue binding

Test Branch 2 – Query Control Interface

- 2A – Query Control Client & Server using SOAP over HTTP binding
- 2B - Query Control Client & Server using XML over AS2 binding

Test Branch 3 – Query Callback Interface

- 3A – Query Callback Sender & Receiver using XML over HTTP binding
- 3B - Query Callback Sender & Receiver using XML over HTTPS binding
- 3C - Query Callback Sender & Receiver using XML over AS2 binding

In the Test Branch column of the Mandatory Requirements Matrix below, we designate the applicable branch for each requirement.

1.1 Mandatory Requirements Matrix

The following table outlines the mandatory requirements for an EPCIS implementation as defined by the EPCIS 1.1 Specification.

Note that column (4)- Applies to the specification line with the "SHALL" or "SHALL NOT" nomenclature or applies to the entire paragraph(s) that encapsulates that line.

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M1	7.2	Ability to provide events in all four event types, with data in each standard field	UML Diag.	0	TCR-43
M2		Deleted			
M3	6.3 7.1.2	Ability to provide extension fields ("implementations SHALL provide for extensibility"), where extension fields can contain arbitrary names and arbitrary values	763, 1603 Last four paragraphs of 7.1.2	0	TCR-43 TCR-2
M4		Deleted			
M5	7.4.1	Ignore recordTime when recordTime is presented to the Capture Interface.	1223	0	TCR-43 TCR-3
M6	7.4.1	Provide recordTime for all events retrieved through the Query Interfaces.	1223	1,2,3	TCR-3
M7	7.4.1	Correctly represent the value of eventTimeZoneOffset	1223	1	TCR-2 TCR-39
M8		Replaced by M102			
M9		Replaced by M103			
M10	7.4.3	If the parentID in an Aggregation Event is an EPC, ensure that the EPC is in the "pure identity" URI form for EPCs as specified in TDS 1.9 and later versions	1365	1	TCR-4
M11		Replaced by M102			
M12		Replaced by M103			
M13	7.4.3	In an Aggregation Event, ensure that the parentID is populated when the Action = ADD or DELETE	1365 (table), 1373, 1375,	1	TCR-7

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
			1378, 1380		
M14	7.4.5	If the parentID in a Transaction Event is an EPC, ensure that the EPC is in the "pure identity" URI form for EPCs as specified in TDS 1.3	1462(table)	1	TCR-4
M15		Replaced by M102			
M16		Replaced by M103			
M17	8.1.2	Ensure that the EPCIS Capture Interface accepts each element of the argument list that is a valid EPCISEvent or subtype thereof per the EPCIS 1.1 Spec	1634	1	TCR-6
M18	8.2.7.1	Implement all subscription control methods of the Query Control Interface, including subscribe and poll (note: Verify also SHALL NOT)	1995	2	TCR-8 TCR-9 TCR-10
M19	8.2.7.1	Support the SimpleEventQuery and SimpleMasterDataQuery with all parameters	2005	2, 3	TCR-11
M20	8.2.5	Throw an InvalidURIException for an incorrect dest argument in the subscribe method in EPCIS Query Control Interface	1792 (table)	2	TCR-40
M21	8.2.5	Return Version “v1.1” or “v1.0” for getStandardVersion method in EPCIS Query Control Interface	1792 (table)	2	TCR-39
M22	8.2.5	Ensure correct behavior for getVendorVersion method in the EPCIS Query Control Interface	1792 (table)	2	TCR-1
M23	8.2.5	If a QueryParams instance includes a name/value pair where a value is empty, execute the query as though the query parameter were omitted altogether	1809	2	TCR-41 TCR-42

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M24	8.2.5	Throw a QueryParameterException for poll or subscribe when a parameter required by a query is omitted or is supplied with an empty value	1811	2	TCR-40
M25	8.2.5	Throw a QueryParameterException for poll or subscribe when a parameter is supplied whose name does not correspond to any parameter name defined by the query	1811	2	TCR-40
M26	8.2.5	Throw a QueryParameterException for poll or subscribe when two parameters are supplied with the same name	1811	2	TCR-40
M27	8.2.5	Throw a QueryParameterException for poll or subscribe when any other constraint imposed by the query is violated – e.g.: range of permitted values for a parameter	1811	2	TCR-40
M28	8.2.5.1	Ensure that raise a SubscriptionControlsException if both “schedule” and “trigger” are both specified or both omitted in SubscriptionControls	1833 (table)	2	TCR-40
M29	8.2.5.2	Ensure proper use of recordTime and initialRecordTime for the first execution of a subscription query	1843	2	TCR-41
M30	8.2.5.3	Raise a SubscriptionControlsException if a query schedule field does not conform to the grammar in Section 8.2.5.3	1865	2	TCR-40
M31	8.2.5.3	Ensure that the response to a subscription query is within reasonable timeframe of the time specified in the QuerySchedule for a lightly loaded system under test	1900	2	TCR-40 TCR-41
M32	8.2.5.3	Raise a SubscriptionControlsException if an EPCIS knows that it cannot honor a QuerySchedule without deviating widely from the request – to be	1903	2	TCR-40

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		arranged by the tester and testee at the time of testing			
M33	8.2.5.3	Ensure that the queryName from the call to poll or subscribe is in the QueryResults	1959 (table)	2	TCR-42
M34	8.2.5.3	Ensure that the subscriptionID from the call to subscribe is in the QueryResults for a subscription query	1959 (table)	2	TCR-41
M35	8.2.5.3	Ensure that subscriptionID is omitted in the QueryResults for a poll query	1959 (table)	2	TCR-42
M36		<DELETED>			
M37	8.2.7.1	Support both poll and subscribe for the SimpleEventQuery	1995	2	TCR-11
M38	8.2.7.1	Return the same events from a query as originally captured, subject to authorization, inclusion of recordTime, and conversions to and from an abstract internal representation – but do not need to preserve the order in the fields defined to hold an unordered list	1999	2, 3	TCR-11
M39	8.2.7.1	Raise a QueryParameterException for an unpermitted value in the EQ_action parameter in SimpleEventQuery	2006 (table)	2	TCR-12
M40	8.2.7.1	Ensure correct values for the orderBy parameter in SimpleEventQuery – i.e.: `eventTime, recordTime, quantity, or the fully qualified name of an extension field whose type is Int, Float, Time, or String`	2006 (table)	2	TCR-12
M41	8.2.7.1	Raise a QueryParameterException if the values of orderDirection parameter are not ASC or DESC in a SimpleEventQuery	2006 (table)	2	TCR-12
M42	8.2.7.1	Raise a QueryParameterException if	2006 (table)	2	TCR-12

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		a Query Client specifies both eventCountLimit and maxEventCount in a SimpleEventQuery			
M43	8.2.7.1	Raise a QueryParameterException if the Query Client omits orderBy when specifying eventCountLimit in a SimpleEventQuery	2006 (table)	2	TCR-12
M44	8.2.7.1	Raise a QueryTooLargeException if a query will return more events than a specified maxEventCount in a SimpleEventQuery	2006 (table)	2	TCR-12
M45	8.2.7.2	Raise a SubscribeNotPermittedException when SimpleMasterDataQuery is specified as the queryName argument to subscribe	2051	2	TCR-12
M46	8.2.7.2	Ignore the value of attributeNames if includeAttributes is false in SimpleMasterDataQuery	2054 (table)	2	TCR-13
M47	8.2.7.2	Raise a QueryTooLargeException if SimpleMasterDataQuery will return more vocabulary elements than the specified maxElementCount	2054 (table)	2	TCR-12
M48	8.2.8	Each time EPCIS executes a standing query per the QuerySchedule, attempt to deliver results to subscriber via Query Callback Interface	2087	3	TCR-19
M49	9.1	Ensure proper namespace use and schema declaration for vendor specific extension attributes	2147, 2155	0,1, 2	TCR-22
M50	9.2	If use the Standard Business Document Header, use the specified set of elements in the tables at Line 2232 & 2236	2232,	2, 3	TCR-28
M51	9.2	If use the Standard Business Document Header, use all fields that are required by the SBDH schema, and use the values for these fields as specified in the SBDH	2241	2, 3	TCR-28
M52	9.5	In the schema for Core Event Types, include a time zone specifier of "Z"	2328	0,1	TCR-23

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		for UTC or an explicit offset from UTC for all values of type xsd:dateTime			
M53	9.5	In the schema for Core Event Types, for any XML element that specifies minOccurs = "0" of type xsd:anyURI, xsd:string, or a type derived from one of those, treat an instance having the empty string as its value in the exactly the same way as if the element were omitted altogether	2332 2331	1	TCR-24
M54	9.7	In the schema for Master Data, include a time zone specifier of "Z" for UTC or an explicit offset from UTC for all values of type xsd:dateTime	3118	2	TCR-14
M55	9.7	In the schema for Master Data, for any XML element that specifies minOccurs = "0" of type xsd:anyURI, xsd:string, or a type derived from one of those, treat an instance having the empty string as its value in the exactly the same way as if the element were omitted altogether	3121	2	TCR-14
M56	9.7	When using the schema for Master Data, treat a <children> list containing zero elements in the same way as if the <children> element were omitted altogether	3219 (XML Schema for Master Data)	2	TCR-14
M57	10.1	If implement message queuing as a capture server, provide one or more message queue endpoints through which a capture client may deliver one or more EPCIS events	3370	1B	TCR-25
M58	10.1	If implement message queuing as a capture client, provide a conformant EPCISDocument or EPCIS Query Document to the capture server	3374	0B	TCR-44
M59	10.1	If implement message queuing as a capture server, accept an EPCISDocument from the queue,	3383 - 3387	1B	TCR-25

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		accepting all events or rejecting all events			
M60	10.2	If implement HTTP as a capture server, provide an HTTP URL through which a capture client may deliver one or more EPCIS events	3404	1A	TCR-26
M61	10.2	If implement HTTP as a capture client, support invoking an HTTP POST operation on the URL provided by the capture server	3406	0A	TCR-45
M62	10.2	If implement HTTP as a capture client, provide a conformant EPCISDocument or EPCIS Query Document to the capture server	3415	0A	TCR-45
M63	10.2	If implement HTTP as a capture server, accept an EPCISDocument, returning a status code for successful or unsuccessful capture	3420, 2421, 3422	1A	TCR-26
M64		Deleted			
M65	11.1	Raise a QueryParameterException if a value element in a poll or subscription query is not in valid syntax for the type required by the query parameter	3461, 3475	2	TCR-12
M66	11.1	In the schema for Core Query Operations, include a time zone specifier of “Z” for UTC or an explicit offset from UTC for all values of type xsd:dateTime	3480	2	TCR-16
M67	11.1	In the schema for Core Query Operations, for any XML element that specifies minOccurs = “0” of type xsd:anyURI or xsd:string, treat an instance having the empty string as its value in the exactly the same way as if the element were omitted altogether	3483	2	TCR-16
M68	11.2	If implement the SOAP/HTTP binding for the Query Control Interface, conform to the WSDL at line 3890	3380-3383	2A	TCR-21

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M69	11.2	If implement the SOAP/HTTP binding for the Query Control Interface, when the Query Server receives a SOAP message that is syntactically invalid per the WSDL at line 3890, raise a ValidationException error or raise a more generic exception provided by the SOAP processor being used	3386-3889	2A	TCR-21
M70	11.3	If implement the AS2 binding for the Query Control Interface, conform to the provisions of Section 11.3	4399-4401	2B	TCR-27
M71	11.3	If implement the AS2 binding for the Query Control Interface, provide an HTTP URL through which the Query Server receives messages from the Query Client in accordance with RFC4130	4405	2B	TCR-27
M72	11.3	If implement the AS2 binding for the Query Control Interface, transmit an XML document whose root element conforms to the EPCISQueryDocument element as defined by the schema in Section 11.1	4406	2B	TCR-29
M73	11.3	If implement the AS2 binding for the Query Control Interface, transmit an XML document where the element immediately nested within the EPCISBody element is one of the elements corresponding to an EPCIS Query Control Interface method request (i.e.: one of Subscribe, Poll, ...)	4409	2B	TCR-30
M74	11.3	If implement the AS2 binding for the Query Control Interface, if a message sent by a Query Client does not conform with the requirements in lines 4405-4411, respond with a ValidationException error	4411-4413	2B	TCR-30

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M75	11.3	If implement the AS2 binding for the Query Control Interface, provide an HTTP URL that the query server will use to deliver a response message	4430	2B	TCR-27
M76	11.3	If implement the AS2 binding for the Query Control Interface, comply with the Requirements listed in the GS1 document "EDIINT AS1 and AS2 Transport Communications Guidelines" (EDICG)	4418	2B	TCR-27
M77	11.3	If implement the AS2 binding for the Query Control Interface, include the Standard Business Document Header within the EPCISHeader element	4422	2B	TCR-31
M78	11.3	If implement the AS2 binding for the Query Control Interface, include within the Standard Business Document Header a unique identifier as the value of the InstanceIdentifier element	4423	2B	TCR-31
M79	11.3	If implement the AS2 binding for the Query Control Interface, respond to each message sent by a query client by delivering a response message to the URL provided by the query client, in accordance with RFC4130	4430	2B	TCR-27
M80	11.3	If implement the AS2 binding for the Query Control Interface, provide a response message that is an XML document whose root element conforms to the EPCISQueryDocument element as defined by the schema in Section 11.1	4431-4432	2B	TCR-32
M81	11.3	If implement the AS2 binding for the Query Control Interface, return the element immediately nested within the EPCISBody element as one of the elements shown in the table starting at line 4435	4433-4435	2B	TCR-32

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M82	11.3	If implement the AS2 binding for the Query Control Interface, return the Standard Business Document Header within the EPCISHeader element	4437	2B	TCR-32 TCR-33
M83	11.3	If implement the AS2 binding for the Query Control Interface, return within the Standard Business Document Header the BusinessScope element containing a Scope element containing a CorrelationInformation element containing a RequestingDocumentInstanceIdentifier element	4438	2B	TCR-33
M84	11.3	If implement the AS2 binding for the Query Control Interface, ensure that the value of the RequestingDocumentInstanceIdentifier element is the value of the InstanceIdentifier element from the Standard Business Document Header of the corresponding request	4442	2B	TCR-33
M85	11.3	If implement the AS2 binding for the Query Control Interface, ensure that the Type subelement within the Scope element is set to EPCISQuery, and the InstanceIdentifier is set to EPCIS	4444, 4445	2B	TCR-33
M86		Deleted			
M87	11.4	For the Query Callback Interface, ensure that the destination URI conforms to the general syntax for URIs as defined in RFC2396	4561	3	TCR-17 TCR-18
M88	11.4.1	For the Query Callback Interface, deliver the response payload as an EPCISQueryDocument whose EPCISBody element contains the	4568	3	TCR-36

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		elements in the table at line 4570			
M89	11.4.1	For the Query Callback Interface, ensure that the queryName and subscriptionID fields of the response payload body contain the queryName and subscriptionID values that were supplied in the call to subscribe that created the standing query	4573	3	TCR-20
M90	11.4.2	For the HTTP binding of the Query Callback Interface, ensure that the syntax for HTTP destination URIs is as defined in RFC2616	4578	3A	TCR-17
M91	11.4.2	For the HTTP binding of the Query Callback Interface, ensure that if the port is omitted, the port defaults to 80	4585-4587	3A	TCR-17
M92	11.4.2	For the HTTP binding of the Query Callback Interface, ensure that the Query Server delivers query results by sending an HTTP POST request to the receiver designated in the URL, where remainder-of-URL is included in the HTTP request-line (as defined in RFC2616), and where the payload is an XML document as specified in Section 11.4.1	4589	3A	TCR-17
M93	11.4.2	For the HTTP binding of the Query Callback Interface, ensure that the Query Client interprets a response code of 200-299 as a normal response, not indicative of any error	4593-4596	3A	TCR-37
M94	11.4.3	For the HTTPS binding of the Query Callback Interface, ensure that the syntax for HTTP destination URIs is as defined in RFC2818, Section 2.4	4601	3B	TCR-18
M95	11.4.3	For the HTTPS binding of the Query Callback Interface, deliver query results by sending an HTTP POST request to receiver designated in the URL, where remainder-of-URL is	4614	3B	TCR-18

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		included in the HTTP request-line (as defined in RFC2616), and where the payload is an XML document as specified in Section 11.4.1			
M96	11.4.3	For the HTTPS binding of the Query Callback Interface, use HTTP over TLS as defined in RFC2818. Implement TLS as defined in RFC2246 except that the mandatory cipher suite is TLS_RSA_WITH_AES_128_CBC_SHA, as defined in RFC3268 with CompressionMethod.null	4618, 4619	3B	TCR-38
M97	11.4.3	For the HTTPS binding of the Query Callback Interface, ensure that the Query Client interprets a response code of 200-299 as a normal response, not indicative of any error	4623-4626	3B	TCR-37
M98	11.4.4	For the AS2 binding of the Query Callback Interface, ensure that the syntax for the AS2 destination URI is as2:remainder-of-URI, where remainder-of-URI identifies a specific AS2 communication profile to be used by the EPCIS Service to deliver information to the subscriber. Ensure that the URI syntax complies with RFC2396.	4630	3C	TCR-34
M99	11.4.4	For the AS2 binding of the Query Callback Interface, deliver query results by sending an AS2 message in accordance with RFC4130	4648	3C	TCR-27
M100	11.4.4	For the AS2 binding of the Query Callback Interface, ensure that the AS2 message payload is an XML document as specified in Section 11.4.1	4649	3C	TCR-35
M101	11.4.4	For the AS2 binding of the Query Callback Interface, ensure that the EPCIS Service and recipient of standing query results comply with the requirements listed in EDICG	4651	3C	TCR-27
M102	7.3.3.2	In the “what” dimension of an	918-919	1	TCR-5

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		EPCIS event, the value of an `epc` element SHALL be a URI [RFC2396] denoting the unique instance-level identity for an object.			
M103	7.4.3	The identifier of the parent of the association. When the parent identifier is an EPC, this field SHALL contain the "pure identity" URI for the EPC as specified in [TDS1.9], Section 7.	1365 (table) Aggregation Event	1	TCR-4
M104	7.3.3.3	The `quantity` may be omitted to indicate that the quantity is unknown or not specified. If `quantity` is omitted, then `uom` SHALL be omitted as well.	925 (table) Quantity Field	1	TCR-49
M105	7.3.3.3	If the `QuantityElement` lacks a `uom` field (below), then the quantity SHALL have a positive integer value, and denotes a count of the number of instances of the specified `EPCClass` that are denoted by this `QuantityElement`.	925 (table)	1	TCR-49
M106	7.3.3.3	If the `QuantityElement` includes a `uom`, then the quantity SHALL have a positive value (but not necessarily an integer value), and denotes the magnitude of the physical measure that specifies how much of the specified `EPCClass` is denoted by this `QuantityElement`	925 (table) Quantity Field	1	TCR-49
M107	7.3.3.3	Uom: If present, specifies a unit of measure by which the specified quantity is to be interpreted as a physical measure, specifying how much of the specified `EPCClass` is denoted by this `QuantityElement`. The `uom` SHALL be omitted if `quantity` is omitted.	925 (table) Uom Field	1	TCR-49

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M108	7.3.3.3.1	When a `uom` field is present, its value SHALL be the 2- or 3-character code for a physical unit specified in the "Common Code" column of UN/CEFACT Recommendation 20 [CEFACT20].	945	1	TCR-49
M109	7.3.3.3.1	The code SHALL be a code contained in a row of [CEFACT20] meeting all of the following criteria: • The "Quantity" column contains one of the following quantities: *length*, *area*, *volume*, or *mass*. • The "Status" column does *not* contain "X" (deleted) or "D" (deprecated).	947	1	TCR-49
M110	7.3.3.3.2	When a Vocabulary Element in `EPCClass` represents the class of SGTIN EPCs denoted by a specific GTIN, it SHALL be a URI in the following form, as defined in Version 1.3 and later of the EPC Tag Data Standards	957-959	1	TCR-47
M111	7.3.3.3.3	When a Vocabulary Element in `EPCClass` represents the class of SGTIN EPCs denoted by a specific GTIN and batch/lot, it SHALL be a URI in the following form, as defined in [TDS1.9, Section 6]	969-971	1	TCR-46 TCR-47
M112	7.4.2	An `ObjectEvent` SHALL NOT contain `ilmd` if `action` is `OBSERVE` or `DELETE`.	1263 (table)	1	TCR-51
M113	7.4.3	An AggregationEvent SHALL contain either a non-empty `childEPCs`, a non-empty `childQuantityList`, or both, except that both `childEPCs` and `childQuantityList` MAY be empty if `action` is `DELETE`, indicating that all children are disaggregated from the parent.	1365 (table) Aggregation Event	1	TCR-46 TCR-47

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
M114	7.4.5	A TransactionEvent SHALL contain either a non-empty `epcList`, a non-empty `quantityList`, or both, except that both `epcList` and `quantityList` MAY be empty if `action` is `DELETE`, indicating that all the objects are disassociated from the business transaction(s).	1462 (table)	1	TCR-46
M115	7.4.6	If `transformationID` is omitted, then a `TransformationEvent` SHALL include at least one input (i.e., at least one of `inputEPCList` and `inputQuantityList` are non-empty) AND at least one output (i.e., at least one of `outputEPCList` and `outputQuantityList` are non-empty).	1539	1	TCR-48
M116	7.4.6	If `transformationID` is included, then a `TransformationEvent` SHALL include at least one input OR at least one output (or both).	1542	1	TCR-48
M117	8.2.7.1.1	The parameter list for `MATCH_epc`, `MATCH_parentID`, `MATCH_inputEPC`, `MATCH_outputEPC`, and `MATCH_anyEPC` SHALL be processed as follows. Each element of the parameter list may be a pure identity pattern as specified in [TDS1.9], or any other URI.	2027-2028	2	TCR-52
M118	8.2.7.1.1	The parameter list for `MATCH_epcClass`, `MATCH_inputEPCClass`, `MATCH_outputEPCClass`, and `MATCH_anyEPCClass` SHALL be processed as follows. Let *P* be one of the patterns specified in the value for this parameter, and let *C* be the value of an `epcClass` field in	2033-2034	2	TCR-52

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Test Branch	How Verified (by Demonstration or by Design)
		the appropriate quantity list of an event being considered for inclusion in the result.			
M119	9.1	Replace the `<xsd:any … namespace="##local"/>` declaration with (a) new elements (which SHALL NOT be in any namespace; equivalently, which SHALL be in the empty namespace); followed by (b) a new `extension` element whose type is constructed as described before.	2195-2196	2	TCR-22

1.2 Optional Requirements Matrix

The following table outlines those functional requirements that are defined as optional by the EPCIS 1.1 Specification

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Area	How Verified (by Demonstration or by Design)
O1	7.3.3.1	For all events and operations EPCIS provides for objects to be identified in two ways: Instance-level An identifier is said to be an instance-level identifier if such identifiers are assigned so that each is unique to a single object. That is, no two objects are allowed to carry the same instance-level identifier. Class-level An identifier is said to be a class-level identifier if multiple objects may carry the same identifier.	~~898~~ 897		TCR-46
O2	7.3.3.3	A QuantityElement is a structure that identifies a specific quantity of objects identified by a specific class-level identifier. It has the following structure: QuantityElement epcClass(EPCClass) quantity(Float) uom(UOM(fixed|variable)) Needs to be supported for all events and operations.	924		TCR-49
O3	7.3.5.4	A Source or Destination is used to provide additional business context when an EPCIS event is part of a business transfer; That is, a process in which there is a transfer of ownership, responsibility, and/or custody of physical or digital objects.	1147		TCR-50
O4	7.3.5.4	SourceDestTypeID : An identifier that indicates what kind of source or destination this Source or Destination (respectively) denotes.	1163 (table)		TCR-50
O5	7.3.5.4	SourceDestID: An identifier that denotes a specific source or destination.	1163 (table)		TCR-50
O6	7.3.6	Instance/Lot Master Data (ILMD) may only be included in ObjectEvents with action ADD, and in TransformationEvents. In the case of a TransformationEvent, ILMD applies to the outputs of the transformation, not the inputs. The structure of ILMD defined in this EPCIS standard consists of a set of named attributes,	1209		TCR-51

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Area	How Verified (by Demonstration or by Design)
		with values of any type.			
O7	7.4.3	The AggregationEvent type includes fields that refer to a single "parent" (often a "containing" entity) and one or more "children" (often "contained" objects). A parent identifier is required when `action` is `ADD` or `DELETE`, but optional when `action` is `OBSERVE`. The AggregationEvent is intended to indicate aggregations among objects, and so the children are identified by EPCs and/or EPC classes.	1339 1348		TCR-47
O8	7.4.2	quantityList : (Optional) An unordered list of one or more QuantityElements identifying (at the class level) objects to which the event pertained. An ObjectEvent SHALL contain either a non-empty epcList, a non-empty quantityList, or both.	1263 (table)		TCR-47
O9	7.4.3	childQuantityList: (Optional) An unordered list of one or more QuantityElements identifying (at the class level) contained objects. An AggregationEvent SHALL contain either a non-empty childEPCs, a non-empty childQuantityList, or both, except that both childEPCs and childQuantityList MAY be empty if action is DELETE, indicating that all children are disaggregated from the parent.	1365 (table)		TCR-47
O10	7.4.6	inputEPCList: (Optional) An unordered list of one or more EPCs identifying (at the instance level) objects that were inputs to the transformation.	1537 (table)		TCR-48
O11	7.4.6	inputQuantityList: (Optional) An unordered list of one or more QuantityElements identifying (at the class level) objects that were inputs to the transformation.	1537 (table)		TCR-48
O12	7.4.6	outputEPCList : (Optional) An unordered list of one or more EPCs naming (at the instance level) objects that were outputs from the transformation.	1537 (table)		TCR-48

Req. #	Protocol SubClause	Requirements	Applies to Spec Line # or a containing paragraph(s)	Area	How Verified (by Demonstration or by Design)
O13		outputQuantityList : (Optional) An unordered list of one or more QuantityElements identifying (at the class level) objects that were outputs from the transformation.	1537 (table)		TCR-48
O14	7.4.6	transformationID : (Optional) A unique identifier that links this event to other TransformationEvents having an identical value of transformationID. When specified, all inputs to all events sharing the same value of the transformationID may contribute to all outputs of all events sharing that value of transformationID. If transformationID is omitted, then the inputs of this event may contribute to the outputs of this event, but the inputs and outputs of other events are not connected to this one.	1537 (table)		TCR-48

128

2 Test Case Requirements

An EPCIS Conformance Certification Program will test an Implementation Under Test (IUT) according to predefined test case requirements that have been designed to isolate and test specific features and functions of the EPCIS 1.1 Specification. While these test case requirements are not exhaustive, they test all the mandatory features that are required by the specification.

2.1 Test Case Requirement 1 – Get Version

Get Version		
TPId: TCR-1		
Requirement Purpose: This Test Case Requirement confirms the proper functions of the Query Interface methods that return the EPCIS standard version and the vendor version for the EPCIS Query server under test. The return of correct version numbers also confirms the correct implementation is being tested. **Requirements Tested**: M21, M22		
Pre-test conditions: • The EPCIS Query server is running. • An implementation supported binding is selected: XML over AS2 or SOAP over HTTP (WSDL)		
Step	**Step description**	**Expected results**
1	Invoke the Query Control Interface getStandardVersion method	Confirm the string "1.1" is returned.
2	Invoke the Query Control Interface getVendorVersion method.	Confirm that either an empty string or a string conforming to a proper URI is returned. The possible values of this string and their meanings are vendor-defined, except that: • The empty string SHALL indicate that the implementation implements only standard functionality with no vendor extensions. • When an implementation chooses to return a non-empty string, the value returned SHALL be a URI where the vendor is the owning authority.
3	Repeat steps 1 and 2 for the other binding (if supported by the implementation)	

2.2 Test Case Requirement 2 – Capture of events with all event types, with extensions, and with eventTimeZoneOffset

Capture of events with all event types, with extensions, and with eventTimeZoneOffset		
TPId: TCR-2		
Requirement Purpose: This Test Case Requirement confirms that an implementation can capture events with all event types with data in each standard field, with extension fields, and with the eventTimeZoneOffset field **Requirements Tested**: M2, M4, M7		
Pre-test conditions: • Create a set of events that contains all event types with data in each standard field, multiple extension fields, and the eventTimeZoneOffset field. • Otherwise the EPCIS system should not contain any EPCIS events		
Step	**Step description**	**Expected results**
1	Capture the set of events.	Confirm that the system contains the captured events – including all event types with data in each standard field, multiple extension fields, and the eventTimeZoneOffset field

2.3 Test Case Requirement 3 – Event recordTime in Capture and Query

Event recordTime in Capture and Query		
TPId: TCR-3		
Requirement Purpose: Ensure recordTime is handled correctly during capture and provided correctly during query. Ignore recordTime when recordTime is presented to the Capture Interface. Provide recordTime for all events retrieved through the Query Interfaces. **Requirements Tested**: M5, M6		
Pre-test conditions: • Entries do not exist in the EPCIS Repository for the EPCISEvents used during any capture tests. • Entries exist in the EPCIS Repository for EPCISEvents used in the query test steps. These should be the same events used during the capture test if the software under test implements both capture and query interfaces. If they are not the same EPCISEvents, ensure that recordTime is stored in the repository and the stored value is known for each EPCISEvent to be queried. • Select one binding to test, either AS2 or Web Services, and use the same binding for all test steps until directed to switch. The initial binding chosen is binding A and the other is referred to as binding B in the test steps.		
Step	**Step description**	**Expected results**
1	Skip to step 19 if a capture interface is not under test.	N/A
2	Submit a valid ObjectEvent with an Action of ADD to the capture interface with recordTime set to 2006-11-29T18:00:00Z.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
3	Submit a valid ObjectEvent with a different eventTime or epcList and an Action of ADD to the capture interface with recordTime omitted.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
4	Submit the ObjectEvent from step 2 but with a later eventTime and an Action of OBSERVE to the capture interface.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
5	Submit the ObjectEvent from step 3 but with a later eventTime and an Action of OBSERVE to the capture interface.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
6	Submit a valid AggregationEvent with an Action of ADD to the capture interface with recordTime set to 2006-11-29T18:00:00Z.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.

7	Submit a valid AggregationEvent with a different eventTime, parentID, and childEPCs list with an Action of ADD to the capture interface with recordTime omitted.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
8	Submit the AggregationEvent from step 6 but with an Action of OBSERVE and a later eventTime to the capture interface.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
9	Submit the AggregationEvent from step 7 but with an Action of OBSERVE and a later eventTime to the capture interface.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
10	Submit a valid QuantityEvent to the capture interface.	The event should trigger a warning by the EPCIS repository and should raise exception conditions (negative test).
11	Deleted	
12	Submit a valid TransactionEvent which includes epcList, bizLocation, and Action set to ADD to the capture interface with the recordTime set to 2006-11-29T18:00:00Z.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
13	Submit a valid TransactionEvent which includes a different bizTransactionList, epcList and, bizLocation than step 12, with Action set to ADD to the capture interface with the recordTime omitted.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
14	Submit the TransactionEvent from step 12 but with a later eventTime, new bizLocation and Action set to OBSERVE.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
15	Submit the TransactionEvent from step 13 but with a later eventTime, new bizLocation and Action set to OBSERVE.	The event is recorded in the EPCIS repository, with recordTime set to the date and time of the test.
16	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 2-16 using binding B, and modifying EPCs at the CompanyPrefix level during capture to avoid duplicated events in the repository.	As defined in steps 2 – 15.
17	The test is complete. Steps 18 – 23 are only used if a capture interface is not under test.	N/A

18	Invoke the poll EPCIS method of the query interface using SimpleEventQuery as the queryName argument and one minute before step 2 test date and time as the GE_recordTime parameter in the params argument. Omit all other parameters.	All events submitted to the capture interface in steps 2-16 are returned in the QueryResults instance and the recordTime in all events is the date and time the capture test was done.
19	Invoke the poll EPCIS method of the query interface using SimpleEventQuery as the queryName argument. The params argument should include ObjectEvent as the eventType, a known recordTime for an ObjectEvent in the repository as the GE_recordTime parameter, and one minute later than the known recordTime as the LT_recordTime parameter. Omit all other parameters.	QueryResults should include the known ObjectEvent and the recordTime returned should match the recordTime in the repository for that ObjectEvent.
20	Invoke the poll EPCIS method of the query interface using SimpleEventQuery as the queryName argument. The params argument should include AggregationEvent as the eventType, a known recordTime for an AggregationEvent in the repository as the GE_recordTime parameter, and one minute later than the known recordTime as the LT_recordTime parameter. Omit all other parameters.	QueryResults should include the known AggregationEvent and the recordTime returned should match the recordTime in the repository for that AggregationEvent.
21	Deleted	
22	Invoke the poll EPCIS method of the query interface using SimpleEventQuery as the queryName argument. The params argument should include TransactionEvent as the eventType, a known recordTime for a TransactionEvent in the repository as the GE_recordTime parameter, and one minute later than the known recordTime as the LT_recordTime parameter. Omit all other parameters.	QueryResults should include the known TransactionEvent and the recordTime returned should match the recordTime in the repository for that TransactionEvent.
23	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 19 - 22 using binding B.	As defined in steps 19 – 22.

148

149 2.4 Test Case Requirement 4 – EPC has Pure Identity Form

EPC has Pure Identity Form		
TPId: TCR-4		
Requirement Purpose: Ensure EPC strings conform to [TDS1.3] Section 4.1 "Pure Identity URI form" specification during capture interface processing. **Requirements Tested**: M10, M14, M103		
Pre-test conditions: • The software under test has implemented a capture interface. • Entries do not already exist in the EPCIS Repository for the EPCISEvents submitted in these tests. • Select one binding to test, either AS2 or Web Services, and use the same binding for all test steps until directed to switch. The initial binding chosen is binding A and the other is referred to as binding B in the test steps.		
Step	**Step description**	**Expected results**
1	Submit an otherwise valid ObjectEvent with an Action of ADD to the capture interface with all elements in epcList set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
2	Submit the ObjectEvent from step 1 to the capture interface again, but with a later eventTime and an Action of OBSERVE.	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
3	Submit the ObjectEvent from step 1 to the capture interface again, but with a later eventTime and an Action of DELETE.	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
4	Submit an otherwise valid AggregationEvent with an Action of ADD to the capture interface with all elements in childEPCs set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.

5	Submit an otherwise valid AggregationEvent with an Action of OBSERVE to the capture interface with all elements in childEPCs set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
6	Submit an otherwise valid AggregationEvent with an Action of DELETE to the capture interface with all elements in childEPCs set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
7	Submit an otherwise valid AggregationEvent with an Action of ADD to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
8	Submit an otherwise valid AggregationEvent with an Action of OBSERVE to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
9	Submit an otherwise valid AggregationEvent with an Action of DELETE to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
10	Submit an otherwise valid TransactionEvent with an Action of ADD to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
11	Submit an otherwise valid TransactionEvent with an Action of OBSERVE to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.

12	Submit an otherwise valid TransactionEvent with an Action of DELETE to the capture interface, but with a parentID set to an EPC which is not a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
13	Submit an otherwise valid TransactionEvent with an Action of ADD to the capture interface with all elements in epcList set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
14	Submit an otherwise valid TransactionEvent with an Action of OBSERVE to the capture interface with all elements in epcList set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
15	Submit an otherwise valid TransactionEvent with an Action of DELETE to the capture interface with all elements in epcList set to valid URI's as defined in [RFC2396], but with one or more of those elements set to EPCs that are not "pure identity" URI's as specified in [TDS1.3], Section 4.1.	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
16	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 1 – 15 using interface B.	As defined in steps 1 – 15.

150

151

152

153 2.5 Test Case Requirement 5– URI conforms to RFC2396

URI Conforms to RFC2396		
TPId: TCR-5		
Requirement Purpose: Ensure URI strings conform to RFC2396 definition during capture interface processing. **Requirements Tested**: M102		
Pre-test conditions: • The software under test has implemented a capture interface. • Entries do not already exist in the EPCIS Repository for the EPCISEvents submitted in these tests. • Select one binding to test, either AS2 or Web Services, and use the same binding for all test steps until directed to switch. The initial binding chosen is binding A and the other is referred to as binding B in the test steps.		
Step	**Step description**	**Expected results**
1	Submit an otherwise valid ObjectEvent with an Action of ADD to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
2	Submit an otherwise valid ObjectEvent with an Action of OBSERVE to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
3	Submit an otherwise valid ObjectEvent with an Action of DELETE to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The ObjectEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
4	Submit an otherwise valid AggregationEvent with an Action of ADD to the capture interface with one or more elements in childEPCs set to URI's which are not valid as defined in [RFC2396].	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
5	Submit an otherwise valid AggregationEvent with an Action of OBSERVE to the capture interface with one or more elements in childEPCs set to URI's which are not valid as defined in [RFC2396].	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.

6	Submit an otherwise valid AggregationEvent with an Action of DELETE to the capture interface with one or more elements in childEPCs set to URI's which are not valid as defined in [RFC2396].	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
7	Submit an otherwise valid TransactionEvent with an Action of ADD to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
8	Submit an otherwise valid TransactionEvent with an Action of OBSERVE to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
9	Submit an otherwise valid TransactionEvent with an Action of DELETE to the capture interface with one or more elements in epcList set to URI's which are not valid as defined in [RFC2396].	The TransactionEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
10	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 1 – 9 using interface B.	As defined in steps 1 – 9.

154

155

156

157 2.6 Test Case Requirement 6 – Valid EPCISEvent or Subtype

Valid EPCISEvent or Subtype		
TPId: TCR-6		
Requirement Purpose: Ensure that the EPCIS Capture interface accepts each element of the argument list that is a valid EPCISEvent or subtype thereof. **Requirements Tested**: M17		
Pre-test conditions: • The software under test has implemented a capture interface. • Entries do not already exist in the EPCIS Repository for the EPCISEvents submitted in these tests. • Select one binding to test, either AS2 or Web Services, and use the same binding for all test steps until directed to switch. The initial binding chosen is binding A and the other is referred to as binding B in the test steps.		
Step	**Step description**	**Expected results**
1	Submit a valid ObjectEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.9 and with Action set to ADD. Add 4 lines excluding required fields.	The event is recorded in the EPCIS repository, and all fields are present.
2	Submit a valid ObjectEvent to the capture interface which *excludes* one or more required fields as specified in [EPCIS1.1], Section 7.2.9 and with Action set to ADD.	The event will not be recorded.
3	Submit a valid ObjectEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.9 and with Action set to OBSERVE.	The event is recorded in the EPCIS repository and all fields are present.
4	Submit a valid ObjectEvent to the capture interface which excludes one or more required fields as specified in [EPCIS1.1], Section 7.2.9 and with Action set to OBSERVE.	The event will not be recorded.
5	Submit a valid AggregationEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.10 and with Action set to ADD.	The event is recorded in the EPCIS repository and all fields are present.

6	Submit a valid AggregationEvent to the capture interface which *excludes* one or more required fields as specified in [EPCIS1.1], Section 7.2.10 and with Action set to ADD.	The event will not be recorded.
7	Submit a valid AggregationEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.10 and with Action set to OBSERVE.	The event is recorded in the EPCIS repository and all fields are present.
8	Submit a valid AggregationEvent to the capture interface which *excludes* one or more all required fields as specified in [EPCIS1.1], Section 7.2.10 and with Action set to OBSERVE.	The event will not be recorded.
9	Deleted	
10	Deleted	
11	Deleted	
12	Deleted	
13	Submit a valid TransactionEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.12 and with Action set to ADD.	The event is recorded in the EPCIS repository and all fields are present.
14	Submit a valid TransactionEvent to the capture interface which *excludes* one or more required fields as specified in [EPCIS1.1], Section 7.2.12 and with Action set to ADD.	The event will not be recorded.
15	Submit a valid TransactionEvent to the capture interface which includes all required and optional fields as specified in [EPCIS1.1], Section 7.2.12 and with Action set to OBSERVE.	The event is recorded in the EPCIS repository and all fields are present.
16	Submit a valid TransactionEvent to the capture interface which *excludes* one or more required fields as specified in [EPCIS1.1], Section 7.2.12 and with Action set to OBSERVE.	The event will not be recorded.
17	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 1 – 8 using interface B.	As defined in steps 1 – 8.

158

159

2.7 Test Case Requirement 7 – parentID is Populated for ADD or DELETE Actions in Aggregation Events

parentID is Populated for ADD or DELETE Actions in Aggregation Events		
TPId: TCR-7		
Requirement Purpose: Ensure that aggregation events contain a parentID for ADD or DELETE actions. **Requirements Tested**: M13		
Pre-test conditions: • The software under test has implemented a capture interface. • Entries do not already exist in the EPCIS Repository for the EPCISEvents submitted in these tests. • Select one binding to test, either AS2 or Web Services, and use the same binding for all test steps until directed to switch. The initial binding chosen is binding A and the other is referred to as binding B in the test steps.		
Step	**Step description**	**Expected results**
1	Submit an otherwise valid AggregationEvent with three EPC's included in childEPCs and Action of ADD to the capture interface, omitting parentID.	The AggregationEvent is not recorded. Any other EPCISEvents within the EPCISDocument are not recorded.
2	Submit the AggregationEvent from step 1 with to the capture interface, but with parentID set to a "pure identity" URI as specified in [TDS1.3], Section 4.1.	The AggregationEvent is recorded properly.
3	Submit the AggregationEvent from step 1 with an Action of DELETE to the capture interface.	The AggregationEvent is ignored. Any other EPCISEvents within the EPCISDocument are not recorded.
4	Submit the AggregationEvent from step 2, but changing the Action to DELETE.	The AggregationEvent is recorded properly.
5	If both AS2 and Web Services bindings are implemented in the software under test, repeat steps 1 - 4 using binding B.	As defined in steps 1- 4.

2.8 Test Case Requirement 8 – Query Methods for Subscribe

Query Methods for Subscribe		
TPId: TCR-8		
Requirement Purpose: This Test Case Requirement confirms that an implementation can support the query controls for "subscribe", "getSubscriptionIDs", and "unsubscribe" in the EPCIS Query Control Interface. Requirements Tested: M18		
Pre-test Note: **Pre-test conditions**: • Capture a set of EPCIS standard events		
Step	**Step description**	**Expected results**
1	If the system supports the SOAP over HTTP (WSDL) binding, use the SOAP over HTTP (WSDL) binding to create a SimpleEventQuery with the subscribe control. Populate the queryName, query parameters, the dest URI, subscription controls, and a subscriptionID. Ensure that the query parameters match events within the set of captured EPCIS events.	None
2	If the system supports the HTTP binding, setup the HTTP binding for the Query Callback Interface	None
3	Run the getSubcriptionIDs query operation	Confirm that the system returns the subscriptionID created in Step 1
4	Set the system time to cause the subscription to run	Confirm that the system returns the EPCIS events over HTTP to the dest URI that match the query parameters created in Step 1
5	If the system supports the HTTPS binding, setup the HTTPS binding for the Query Callback Interface	None
6	Set the system time to cause the subscription to run	Confirm that the system returns the EPCIS events over HTTPS to the dest URI that match the query parameters created in Step 1
7	Run the unsubscribe query operation for the subscriptionID created in Step 1	None
8	Set the system time to cause the subscription to run	Confirm that the system returns no events

9	If the system supports the AS2 binding, use the AS2 binding to create a SimpleEventQuery with the subscribe control. Populate the queryName, query parameters, the dest URI, subscription controls, and a subscriptionID. Ensure that the query parameters match events within the set of captured EPCIS events.	None
10	Setup the AS2 binding for the Query Callback Interface	None
11	Run the getSubscriptionIDs query operation	Confirm that the system returns the subscriptionID created in Step 9
12	Set the system time to cause the subscription to run	Confirm that the system returns the EPCIS events over HTTP to the dest URI that match the query parameters created in Step 9
13	Run the unsubscribe query operation for the subscriptionID created in Step 9	None
14	Set the system time to cause the subscription to run	Confirm that the system returns no events

165
166

2.9 Test Case Requirement 9 – Get Query Names

Get Query Names		
TPId: TCR-9		
Requirement Purpose: This Test Case Requirement confirms the proper functioning of the EPCIS method that returns the EPCIS standard query names. **Requirements Tested**: M18		
Pre-test conditions: • None		
Step	**Step description**	**Expected results**
1	If the system supports the SOAP over HTTP (WSDL) binding, use the SOAP over HTTP (WSDL) query binding to invoke the getQueryNames method	Confirm that the system returns the strings "SimpleEventQuery" and "SimpleMasterDataQuery"
2	If the system supports the AS2 binding, use the AS2 query binding to invoke the getQueryNames method	Confirm that the system returns the strings "SimpleEventQuery" and "SimpleMasterDataQuery"

2.10 Test Case Requirement 10 – Query Methods for Poll

Query Methods for Poll		
TPId: TCR-10		
Requirement Purpose: This Test Case Requirement confirms that an implementation can support the poll command for the SimpleEventQuery and for the SimpleMasterDataQuery **Requirements Tested**: M18		
Pre-test conditions: • Capture a set of EPCIS standard events • Load a set of master data into the EPCIS		
Step	**Step description**	**Expected results**
1	If the system supports the SOAP over HTTP (WSDL) binding, use the SOAP over HTTP (WSDL) query binding to run a SimpleEventQuery with the poll control. Populate the queryName and query parameters. Ensure that the query parameters match events within the set of captured EPCIS events.	Confirm that the system returns the events specified by the query parameters
2	Using the SOAP over HTTP (WSDL) query binding, run a SimpleMasterDataQuery with the poll control. Populate the queryName and query parameters. Ensure that the query parameters match master data within the set of loaded master data.	Confirm that the system returns the master data specified by the query parameters.
3	If the system supports the AS2 binding, use the AS2 query binding to run a SimpleEventQuery with the poll control. Populate the queryName and query parameters. Ensure that the query parameters match events within the set of captured EPCIS events.	Confirm that the system returns the events specified by the query parameters
4	Using the AS2 query binding, run a SimpleMasterDataQuery with the poll control. Populate the queryName and query parameters. Ensure that the query parameters match master data within the set of loaded master data.	Confirm that the system returns the master data specified by the query parameters.

2.11 Test Case Requirement 11 – Parameters for SimpleEventQuery and SimpleMasterDataQuery

Parameters for SimpleEventQuery and SimpleMasterDataQuery
TPId: TCR-11
Requirement Purpose: This Test Case Requirement confirms the correct operation the SimpleEventQuery and SimpleMasterDataQuery with all parameters. Second, confirms to support subscribe for SimpleEventQuery. Also, each event of result that is selected to be returned SHALL be identical to the originally event. **Requirements Tested**: M19, M37, M38

Pre-test conditions:

- Capture a set of EPCIS standard events (M1,M2).
- Load a set of master data into the EPCIS

Parameter Set A					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
vocabularyName	A	includeAttributes	true	includeChildren	true
Parameter Set B					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
attributeNames	A	includeAttributes	true	includeChildren	true
Parameter Set C					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
EQ_name	N	includeAttributes	true	includeChildren	true
Parameter Set D					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
WD_name	N	includeAttributes	true	includeChildren	false
Parameter Set E					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
HASATTR	N	includeAttributes	false	includeChildren	true
Parameter Set F					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
EQATTR_attrname	N(attrname is M)	includeAttributes	true	includeChildren	true
Parameter Set G					
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
maxEventCount	N	includeAttributes	false	includeChildren	False

Subscription Control Set H					
Argument	**Value**	**Argument**	**Value**	**Argument**	**Value**
schedule	second : 0	initialRecordTime	A	reportIfEmpty	false

Step	Step description	Expected results
1	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “eventType” and values are N list.	Confirm that the only events with eventType from M1 is that has N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
2	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “GE_eventTime” and value is N.	Confirm that the only events with eventTime from M1 greater than or equal to N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
3	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “LT_eventTime” and value is N.	Confirm that the only events with eventTime from M1 less than N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
4	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “GE_recordTime” and value is N.	Confirm that the only events with recordTime from M1 greater than or equal to N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
5	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “LT_recordTime” and value is N.	Confirm that the only events with recordTime from M1 less than N return are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
6	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “EQ_action” and values are N list.	Confirm that the only events with action from M1 that the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
7	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “EQ_bizStep” and values are N list.	Confirm that the only events with bizStep from M1 that the value matches one of N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
8	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “EQ_disposition” and values are N list.	Confirm that the only events of result with disposition from M1 that the value matches one of N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
9	Invoke Poll using the queryName “SimpleEventQuery”. The parameter Name is “EQ_readPoint” and values are N list.	Confirm that the only events with readPoint from M1 that the value matches one of N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.

10	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "WD_readPoint" and values are N list.	Confirm that the only events with readPoint from M1 that the value matches one of N list are returned, or is a direct or indirect descendant of one of the N list. Each event that is selected to be returned SHALL be identical to the original event of M1.
11	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQ_bizLocation" and values are N list.	Confirm that the only events with bizLocation from M1 that the value matches one of N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
12	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "WD_bizLocation" and values are N list.	Confirm that the only events with readPoint from M1 that the value matches one of N list are returned, or is a direct or indirect descendant of one of the N list. Each event that is selected to be returned SHALL be identical to the original event of M1.
13	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQ_bizTransaction_type"(type is M) and values are N list.	Confirm that the only events with bizTransactionList from M1 include the entry whose type is M and the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
14	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "MATCH_epc" and values are N list.	Confirm that the only events with an epcList or a childEPCs from M1 that the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
15	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "MATCH_parentID" and values are N list.	Confirm that the only events with parentID from M1 that the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
16	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "MATCH_anyEPC" and values are N list.	Confirm that the only events with epcList, childEPCs, or parentID from M1 that the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
17	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "MATCH_epcClass" and values are N list.	Confirm that the only events with epcClass from M1 that the value matches one of the N list are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.

18	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQ_quantity" and value is N.	Confirm that the only events of result with quantity from M1 equal to N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
19	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "GT_quantity" and value is N.	Confirm that the only events of result with quantity from M1 greater than N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
20	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "GE_quantity" and value is N.	Confirm that the only events of result with quantity from M1 greater than or equal to N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
21	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "LT_quantity" and value is N.	Confirm that the only events of result with quantity from M1 less than N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
22	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "LE_quantity" and value is N.	Confirm that the only events of result with quantity from M1 less than or equal to N are returned. Each event that is selected to be returned SHALL be identical to the original event of M1.
23	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQ_fieldname"(fieldname is M) and values are N list.	Confirm that the only events of result with extension field whose name is M and value matches one of the N list. Each event that is selected to be returned SHALL be identical to the original event of M2.
24	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQ_fieldname" (fieldname is M) and value is N.	Confirm that the only events of result with extension field whose name is M and value is equal to N. Each event that is selected to be returned SHALL be identical to the original event of M2.
25	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "GT_fieldname" (fieldname is M) and value is N.	Confirm that the only events of result with extension field whose name is M and value is greater than N. Each event that is selected to be returned SHALL be identical to the original event of M2.
26	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "GE_fieldname" (fieldname is M) and value is N.	Confirm that the only events of result with extension field whose name is M and value is greater than or equal to N. Each event that is selected to be returned SHALL be identical to the original event of M2.
27	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "LT_fieldname" (fieldname is M) and value is N.	Confirm that the only events of result with extension field whose name is M and value is less than N. Each event that is selected to be returned SHALL be identical to the original event of M2.

28	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "LE_fieldname" (fieldname is M) and value is N.	Confirm that the only events of result with extension field whose name is M and value is less than or equal to N. Each event that is selected to be returned SHALL be identical to the original event of M2.
29	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EXISTS_fieldname" (fieldname is M).	Confirm that the only events of result with extension field whose name is M. Each event that is selected to be returned SHALL be identical to the original event of M2.
30	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "HASATTR_fieldname" (fieldname is M) and values are N list.	Confirm that the only events of result with extension field whose name is M and type is vocabulary type and the value is a vocabulary element. Each event that is selected to be returned SHALL be identical to the original event of M2.
31	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "EQATTR_fieldname _attrname" (fieldname is L, attrname is M) and values are N list.	Confirm that the only events of result with extension field whose name is L and type is vocabulary type and have attribute whose name is M and the value of attribute is N. Each event that is selected to be returned SHALL be identical to the original event of M2.
32	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "orderBy" and value is N.	Confirm that the results are ordered by N. Each event that is selected to be returned SHALL be identical to the original event of M2.
33	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "orderDirection" and value is N.	Confirm that the results are ordered by orderDirection N. Each event that is selected to be returned SHALL be identical to the original event of M1.
34	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "eventCountLimit" and value is N.	Confirm that the size of results are less than or equal to N. Each event that is selected to be returned SHALL be identical to the original event of M1.
35	Invoke Poll using the queryName "SimpleEventQuery". The parameter Name is "maxEventCount" and value is N.	When the size of results is greater than N, the QueryTooLargeException occurs. Each event that is selected to be returned SHALL be identical to the original event of M1.
36	Invoke Poll using the queryName "MasterDataQuery" and parmeter set A.	Confirm that the only vocabulary elements drawn from one of N list.
37	Invoke Poll using the queryName "MasterDataQuery" and parameter set B.	Confirm that the only vocabulary attributes whose names match one of the specified names will be included in the results.
38	Invoke Poll using the queryName "MasterDataQuery" and parameter set C.	Confirm that the only vocabulary elements whose names match one of the N list.

39	Invoke Poll using the queryName “MasterDataQuery” and parameter set D.	Confirm that the only vocabulary elements whose names match one of the N list or descendants of vocabulary element. The children element should not be included but attribute element should be included in the result.
40	Invoke Poll using the queryName “MasterDataQuery” and parameter set E.	Confirm that the only vocabulary elements that have attributes whose names match one of the N list. The attribute element should not be included but children element should be included in the result.
41	Invoke Poll using the queryName “MasterDataQuery” and parameter set F.	Confirm that the only vocabulary elements that have attributes whose name is equal to M and value matches one of the N list
42	Invoke Poll using the queryName “MasterDataQuery” and parameter set G.	When the size of results is greater than N, the QueryTooLargeException occurs. The attribute element and children element should not be included.
43	Invoke Poll or Subscribe of Query Control Interface using SOAP over HTTP binding or XML over AS2 binding.(from step 1 to step 42)	The EPCIS 1.0 implementation should return SubscribeResult. Each event that is selected to be returned SHALL be identical to the original event.
44	Invoke callback using HTTP, HTTPS or AS2.(from step 1 to step 35) After about one minute, confirm the result.	The EPCIS 1.0 implementation should send the result of subscription to the destination every one minute. If result is empty, it doesn’t send any result. Each event that is selected to be returned SHALL be identical to the original event.

176

177

178

2.12 Test Case Requirement 12 – SimpleEventQuery and SimpleMasterDataQuery Exceptions

Exceptions		
TPId: TCR-12		
Requirement Purpose: This Test Case Requirement confirms that the EPCIS implementation will raise all exceptions as defined in the EPCIS specification. This covers exceptions raised due to incorrect parameters passed in EPCIS API methods and exceptions raised due to missing or invalid parameters. **Requirements Tested**: M39-M45, M47, M65		
Pre-test conditions: • Capture a set of EPCIS standard events • Load a set of master data into the EPCIS		
Step	**Step description**	**Expected results**
1	Invoke poll, using SimpleEventQuery and an unpermitted EQ_action	Verify that the EPCIS implementation raises a QueryParameterException
2	Invoke poll, using SimpleEventQuery passing incorrect orderBy parameters	Verify that the EPCIS implementation raises a QueryParameterException (specification does not specify exception)
3	Invoke poll, using SimpleEventQuery, using incorrect orderDirection	Verify that the EPCIS implementation raises a QueryParameterException
4	Invoke poll, using SimpleEventQuery, using both eventCountLimit and maxEventCount	Verify that the EPCIS implementation raises a QueryParameterException
5	Invoke poll, using SimpleEventQuery, omit orderBy when specifying eventCountLimit	Verify that the EPCIS implementation raises a QueryParameterException
6	Invoke poll, using SimpleEventQuery, which returns more events than specified in a maxEventCount	Verify that the EPCIS implementation raises a QueryTooLargeException
7	Invoke subscribe, using SimpleMasterDataQuery as the queryName	Verify that the EPCIS implementation raises a SubscribeNotPermittedException
8	Invoke poll, using SimpleMasterDataQuery, which returns more events than specified in a maxElementCount	Verify that the EPCIS implementation raises a QueryTooLargeException
9	Invoke poll, pass an invalid query parameter type syntax	Verify that the EPCIS implementation raises a QueryParameterException
10	Run tests 1-9 with the SOAP over HTTP (WSDL) binding	
11	Repeat tests 1-9 with the AS2 binding	

2.13 Test Case Requirement 13 – includeAttribute for SimpleMasterDataQuery

IncludeAttribute		
TPId: TCR-13		
Requirement Purpose: If includeAttributes is false, attributeNames must be ignored. **Requirements Tested**: M46		
Pre-test conditions: • Load a set of master data into the EPCIS		
Step	**Step description**	**Expected results**
1	Invoke poll, using SimpleMasterDataQuery with includeAttributes as false and specified attributeNames	Verify that the EPCIS ignores attributeNames
2	Run test 1 with the SOAP over HTTP (WSDL) binding	
3	Repeat test 1 with the AS2 binding	

2.14 Test Case Requirement 14 – Master Data Schema Compliance

Master Data Schema		
TPId: TCR-14		
Requirement Purpose: Master Data Schema compliance **Requirements Tested**: M54-56		
Pre-test conditions: • Load a set of master data into the EPCIS		
Step	**Step description**	**Expected results**
1	Post Master Data, include values of type xsd:dateTime	Verify that xsd:dateTime fields include a time zone specifier of "Z" for UTC or an explicit offset from UTC
2	Post Master Data, include elements specifying minOccurs = "0" of type xsd:anyURI, xsd:string, or a type derived from one of those and an empty string as its value	Verify that the instance having the empty string is treated in the exactly the same way as if the element were omitted altogether
3	Post Master Data, include <children> list containing zero elements	Verify that the <children> list is treated in the same way as if the <children> element were omitted altogether
4	Run tests 1-3 with the SOAP over HTTP (WSDL) binding	
5	Repeat tests 1-3 with the AS2 binding	

2.15 Test Case Requirement 15 – Type Checking

TPId: TCR-15		
Requirement Purpose: Validate poll or subscription types against the EPCIS schema. **Requirements Tested**: M64		
Pre-test conditions: • None		
Step	**Step description**	**Expected results**
1	Submit a `poll` or `subscribe` request using an invalid type for each query parameter	Verify that the EPCIS implementation raises a QueryParameterException

2.16 Test Case Requirement 16 – Core Query Operations

Core Query Operations		
TPId: TCR-16		
Requirement Purpose: Validate poll or subscription types against the EPCIS schema. **Requirements Tested**: M66-67		
Pre-test conditions: • Capture a set of EPCIS standard events		
Step	**Step description**	**Expected results**
1	Submit a `poll` or `subscribe` request with a xsd:dateTime field, do not include a time zone specifier of "Z" for UTC or an explicit offset from UTC for all values of type xsd:dateTime	Verify EPCIS implementation raises a QueryParameterException
2	Submit a `poll` or `subscribe` request, include elements specifying minOccurs = "0" of type xsd:anyURI, xsd:string, or a type derived from one of those and an empty string as its value	Verify that the instance having the empty string is treated in the exactly the same way as if the element were omitted altogether

2.17 Test Case Requirement 17 – Http binding vs Query callback interface

Http binding vs Query callback interface		
TPId: TCR-17		
Requirement Purpose: This test case requirement confirms that the HTTP URI that is sent with subscribe request as destination is conforming to its proper format and working properly for HTTP binding. **Requirements Tested**: M90-92, M87		
Pre-test conditions: Query #1 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : wrong formatted HTTP URI(that does not support the format defined in RFC2616) • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n>a Query #2 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : Formatted HTTP URI that does not contain port • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n>b Query #3 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : Formatted HTTP URI that contains port • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n>c		
Step	**Step description**	**Expected results**
1	Client sets up the subscription with Query #1 parameters and subscription ID ‘QueryTCR<n>a’ on Server EPCIS	Server will raise an `InvalidURIException`
2.	Client sets up the subscription with Query #2 parameters and subscription ID ‘QueryTCR<n>b’ on Server EPCIS	Subscribe call will be successful
3.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI and port will be taken as 80.

4.	Invoke Unsubscribe with ID ‘QueryTCR<n>b’	Unsubscribe call will be successful
5.	Client sets up the subscription with Query #3 parameters and subscription ID ‘QueryTCR<n>c’ on Server EPCIS	Subscribe call will be successful
6.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI .
7.	Invoke Unsubscribe with ID ‘QueryTCR<n>c’	Unsubscribe call will be successful

200

201

2.18 Test Case Requirement 18 – HTTPS binding vs Query callback interface

HTTPS binding vs Query callback interface		
TPId: TCR-18		
Requirement Purpose: This test case requirement confirms that the HTTPS URI that is sent with subscribe request as destination conforms to its proper format and working properly for HTTPS binding. **Requirements Tested**: M87, M94, M95		
Pre-test conditions: Query #1 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : wrong formatted HTTPS URI(that does not support the format defined in RFC2818) • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n+1>a Query #2 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : Formatted HTTPS URI • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n+1>b		
Step	**Step description**	**Expected results**
1	Client sets up the subscription with Query #1 parameters and subscription ID 'QueryTCR<n+1>a' on Server EPCIS	Server will raise an `InvalidURIException`
2.	Client sets up the subscription with Query #2 parameters and subscription ID 'QueryTCR<n+1>b' on Server EPCIS	Subscribe call will be successful
3.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI .
4.	Invoke Unsubscribe with ID 'QueryTCR<n+1>b'	Unsubscribe call will be successful

205 ## 2.19 Test Case Requirement 19 – QuerySchedule, delivery of
206 ## results to subscriber via Query Callback Interface

QuerySchedule, delivery of results to subscriber via Query Callback Interface		
TPId: TCR-19		
Requirement Purpose: This test case requirement confirms that Each time EPCIS executes a standing query per the QuerySchedule, attempt to deliver results to subscriber via Query Callback Interface **Requirements Tested**: M48		
Pre-test conditions: Query #1 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : Formatted HTTP URI or HTTPS URI or AS2 URI • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n+2>a		
Step	**Step description**	**Expected results**
1	Client sets up the subscription with Query #1 parameters and subscription ID ‘QueryTCR<n+2>a’ on Server EPCIS	subscribe call will be successful
2.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI .
3.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI .
4.	Invoke Unsubscribe with ID ‘QueryTCR<n+2>a’	Unsubscribe call will be successful

207

208

209

2.20 Test Case Requirement 20 – Query Callback Interface vs queryName and subscriptionID

Query Callback Interface vs queryName and subscriptionID		
TPId: TCR-20		
Requirement Purpose: This test case requirement confirms that For the Query Callback Interface the queryName and subscriptionID fields of the response payload body contain the queryName and subscriptionID values that were supplied in the call to subscribe that created the standing query. **Requirements Tested**: M89		
Pre-test conditions: Query #1 Framework – • QueryName : SimpleEventQuery • Params : any valid params • destination URI : Formatted HTTP URI or HTTPS URI or AS2 URI • schedule : schedule query for once every 30 sec • subscription ID : QueryTCR<n+3>a		
Step	**Step description**	**Expected results**
1	Client sets up the subscription with Query #1 parameters and subscription ID ‘QueryTCR<n+3>a’ on Server EPCIS	subscribe call will be successful
2.	Wait for 30 sec	After 30 seconds some valid result will be sent to the Dest URI and response payload body contain the queryName and subscriptionID values that were supplied in the call to subscribe that created the standing query.
4.	Invoke Unsubscribe with ID ‘QueryTCR<n+3>a’	Unsubscribe call will be successful

2.21 Test Case Requirement 21 – Query Server, SOAP message that is syntactically valid and invalid per the WSDL

Query Server, SOAP message that is syntactically valid and invalid per the WSDL		
TPId: TCR-21		
Requirement Purpose: This test case requirement confirms that when the Query Server receives a SOAP message that is syntactically valid per the WSDL as outlined in Section 11.2 of EPCIS 1.1, it does not raise an exception. Then this requirement confirms that when the Query Server receives a SOAP message that is syntactically invalid per the WSDL as outlined in Section 11.2 of EPCIS 1.1, it raises a ValidationException error or raise a more generic exception provided by the SOAP processor being used. **Requirements Tested**: M68, M69		
Pre-test conditions:		
Step	**Step description**	**Expected results**
1	Implement the SOAP/HTTP binding conforming to the WSDL as outlined in Section 11.2 in the spec	N/A
2	Invoke the methods that are defined in the WSDL	Server and SOAP processor do not raise an exception
3	Invoke a method (like 'XYZ') that is not defined in the WSDL	Server will raise a ValidationException error or raise a more generic exception provided by the SOAP processor being used.

2.22 Test Case Requirement 22 – Capture Event Extension Namespace

Capture Event Extension's Namespace		
TPId: TCR-22		
Requirement Purpose: Confirms use of proper namespace and schema declaration in the capture event xml. **Requirements Tested**: M49		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create and send a valid capture event having an extension element that has a namespace prefix, but the prefix is undefined in the xml (i.e. there is no xmlns attribute for the prefix).	Server does not store the event containing the extension data.
2.	Create and send a valid capture event having an extension element that has a namespace prefix, and the namespace is urn:epcglobal:epcis:xsd:1.	Server does not store the event containing the extension data.
3.	Create and send a valid capture event having an extension element that has a namespace prefix, and the namespace is not urn:epcglobal:epcis:xsd:1, and is a syntactically valid URL, as defined in section 8.1.3 of CBV 1.1 (“General Considerations for HTTP URLs as User Vocabulary Elements”). The URL must be assigned by the owner of the corresponding internet domain, though this will not be verified by TCR-22.	Server stores the event containing the extension data.
4.	Create and send a valid capture event having an extension element that has a namespace prefix, and the namespace is not urn:epcglobal:epcis:xsd:1, and is a syntactically valid URN as defined in section 8.1.2 of CBV 1.1 (“General Considerations for Private or Industry-wide URN as User Vocabulary Elements”). The URN must be assigned by owner of the URN namespace, though this will not be verified by TCR-22.	Server stores the event containing the extension data.

5.	Create and send a valid capture event having an extension element that has a namespace prefix, and the namespace is not urn:epcglobal:epcis:xsd:1, and is a syntactically valid OID URN whose initial path is a Private Enterprise Number assigned to the vendor, though this will not be verified by TCR-22.	Server stores the event containing the extension data.

220

221

2.23 Test Case Requirement 23 – Capture Event Timezone

Capture Event Timezone		
TPId: TCR-23		
Requirement Purpose: Confirms correct interpretation of capture event time values, with regard to their specified timezone. **Requirements Tested**: M52		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create and send a valid capture event with the document header CreationDateAndTime element, the eventTime element, and the creationDate attribute all having a timezone specified as "Z".	Server stores each of the time values as UTC (no offset).
2.	Create and send a valid capture event with the document header CreationDateAndTime element, the eventTime element, and the creationDate attribute all having a timezone specified as a properly formatted numeric offset.	Server stores each of the time values as UTC adjusted by the specified offset.

2.24 Test Case Requirement 24 – Capture Event Empty String

Capture Event Empty String		
TPId: TCR-24		
Requirement Purpose: Confirms that capture event data consisting of an empty string is treated as if the element were omitted. **Requirements Tested**: M53		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create and send two valid Capture Events, one having bizLocation="" and one having no bizLocation element.	The server's stored bizLocation value is the same for the two events.
2.	Repeat the above test for each item in the following list. With each repetition, replace "bizLocation" with the specified item. 1. bizStep 2. bizTransactionList 3. BusinessServiceName 4. ContactTypeIdentifier 5. Description 6. disposition 7. EmailAddress 8. epc 9. eventTimeZoneOffset 10. ExpectedResponseDateTime 11. FaxNumber 12. Identifier 13. LanguageCode 14. parentID 15. readPoint 16. RequestingDocumentInstanceIdentifier 17. ServiceTransaction 18. TelephoneNumber	For each repetition, verify the item's value stored is the same for the two events.

228 2.25 Test Case Requirement 25 – Capture Server Message Queue

Capture Server Message Queue		
TPId: TCR-25		
Requirement Purpose: Confirms that capture events can be sent to the capture server via message queue. **Requirements Tested**: M57,59		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create a capture event consisting of an EPCISDocument having a single EPCISEvent. Send it to the capture server using the server's message queue mechanism.	Server stores the event's data.
2	Create a capture event consisting of an EPCISQueryDocument having a single EPCISEvent in the EventList. Send it to the capture server using the server's message queue mechanism.	Server stores the event's data.
3	Create and send a capture event consisting of an EPCISDocument having ten different EPCISEvents, to the capture server using the server's message queue mechanism.	Server stores all ten events' data.
4	Create a capture event consisting of an EPCISDocument having ten different EPCISEvents, nine of which are valid, and one of which has an invalid required data value. Send the document to the capture server using message queue.	Server stores none of the ten events' data. If message delivery status is available, the status must indicate failure.

229

230

2.26 Test Case Requirement 26 – Capture Server HTTP Input

Capture Server HTTP Input		
TPId: TCR-26		
Requirement Purpose: Confirms that capture events can be sent to the capture server via HTTP POST. **Requirements Tested**: M60,63		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create a capture event consisting of an EPCISDocument having a single EPCISEvent. Send it to the capture server using HTTP POST to the server's specified URL.	Provided the HTTP return status code is 200, the event's data must be stored. When the HTTP return status code is in the range 300 through 599 inclusive, then none of the event's data must be stored.
2	Create a capture event consisting of an EPCISQueryDocument having a single EPCISEvent in the EventList. Send it to the capture server using HTTP POST to the server's specified URL.	Provided the HTTP return status code is 200, the event's data must be stored. When the HTTP return status code is in the range 300 through 599 inclusive, then none of the event's data must be stored.
3	Create and send a capture event consisting of an EPCISDocument having ten different EPCISEvents, to the capture server using HTTP POST protocol, to the server's specified URL.	Provided the HTTP return status code is 200, all ten events' data must be stored. When the HTTP return status code is in the range 300 through 599 inclusive, then none of any of the events' data must be stored.
4	Create a capture event consisting of an EPCISDocument having ten different EPCISEvents, nine of which are valid, and one of which has an invalid required data value. Send the document to the capture server using HTTP POST protocol, to the server's specified URL.	The HTTP return status code must be in the range 300 to 599 inclusive, and none of any of the events' data must be stored.

2.27 Test Case Requirement 27 – AS2 Configuration Setup

AS2 Configuration Setup		
TPId: TCR-27		
Requirement Purpose: This Test Case Requirement establishes that AS2 protocol setup has been properly configured. **Requirements Tested**: M70, M71, M75, M76, M79, M86, M99, M101		
Pre-test conditions:		
Step	**Step description**	**Expected results**
1	Configure the AS2 protocol specific HTTP URL for Query Client to communicate with the Query Server for sending Query Control Interface requests.	Confirm that the URL is available and functioning.
2	Configure the AS2 protocol specific HTTP URL for Query Server to communicate with the Query Client for sending Query Control Interface responses.	Confirm that the URL is available and functioning.
3	Configure the AS2 protocol specific HTTP URL for Query Server to communicate with the Query Callback Implementation for sending Query Callback Interface responses.	Confirm that the URL is available and functioning.
4	Configure the AS2 protocol specific delivery semantics for URLs (end-points) specified in steps 1, 2 and 3.	Confirm that encryption, digital signing, duplicate message detection, message resubmission on failure etc are specified and functioning.
5	Configure the AS2 protocol specific digital certificate features for URLs (end-points) specified in steps 1, 2 and 3.	Confirm that digital certificates including their chains are properly specified and functioning.

2.28 Test Case Requirement 28 – Standard Business Document Header

Standard Business Document Header		
TPId: TCR-28		
Requirement Purpose: This Test Case Requirement confirms the proper usage of the Standard Business Document Header when the `EPCISHeader` element is included in the XML binding of Core Event Types by the EPCIS implementation under test. **Requirements Tested**: M50, M51		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • The EPCIS Header is used in XML bindings of Core Event Types		
Step	**Step description**	**Expected results**
1	Verify that the `HeaderVersion element` is present within the `StandardBusinessDocumentHeader` element within the `EPCISHeader` element.	Confirm the value of the `HeaderVersion` element is "`1.0`".
2	Verify that the `Standard` element is present within the `DocumentIdentification` element within `StandardBusinessDocumentHeader` element.	Confirm the value of `Standard` is "`EPCglobal`".
3	Verify that the `TypeVersion` element is present within the `DocumentIdentification` element within the `StandardBusinessDocumentHeader` element.	Confirm the value of `TypeVersion` is "`1.0`".
4	For an `EPCISDocument` verify that the `DocumentIdentification/Type` element is present.	Confirm the value of `Type` is "`Events`".
5	For an `EPCISMasterData` document verify that the `DocumentationIdentification/Type` element is present.	Confirm the value of `Type` is "`MasterData`".
6	For an `EPCISQueryDocument` used in the request side of binding verify that the `DocumentIdentification/Type` element is present.	Confirm the value of `Type` is "`QueryControl-Request`".

7	For an `EPCISQueryDocument` used in the response side of binding verify that the `DocumentIdentification/Type` element is present.	Confirm the value of `Type` is "`QueryControl-Response`".
8	For an `EPCISQueryDocument` used in the query callback side of binding verify that the `DocumentIdentification/Type` element is present.	Confirm the value of `Type` is "`QueryCallback`".
9	For an `EPCISQueryDocument` used in any other context verify that the `DocumentIdentification/Type` element is present.	Confirm the value of `Type` is "`Query`".
10	Verify that other required elements of SBDH are present.	Confirm that values of required elements of SBDH are properly populated.

239

240 2.29 Test Case Requirement 29 –Query XML in AS2 Binding for 241 Query Control Interface

Query XML in AS2 Binding for Query Control Interface		
TPId: TCR-29		
Requirement Purpose: This Test Case Requirement confirms that all messages sent by the query client to the query server under the AS2 binding of EPCIS QueryControl Interface should be XML documents whose root element is the `EPCISQueryDocument` element. **Requirements Tested**: M72		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • Requirements for TCR-28 are satisfied		
Step	**Step description**	**Expected results**
1	Query Client sends an XML query to Query Server.	Confirm the root element of the XML query is "`EPCISQueryDocument`".

242

243

2.30 Test Case Requirement 30 – AS2 Binding for Query Control Interface

AS2 Binding for Query Control Interface		
TPId: TCR-30		
Requirement Purpose: This Test Case Requirement confirms that all XML documents sent by the query client to the query server under the AS2 binding of EPCIS QueryControl Interface should have an element immediately nested within the EPCISBody element corresponding to an EPCIS Query Control Interface method request. **Requirements Tested**: M73, M74		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • Requirements for TCR-27 are satisfied • Requirements for TCR-29 are satisfied		
Step	**Step description**	**Expected results**
2	Query Client sends a `getStandardVersion` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`GetStandardVersion`".
3	Query Client sends a `getVendorVersion` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`GetVendorVersion`".
4	Query Client sends a `getQueryNames` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`GetQueryNames`".
5	Query Client sends a `Poll` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`Poll`".
6	Query Client sends a `Subscribe` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`Subscribe`"
7	Query Client sends an `Unsubscribe` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`Unsubscribe`".
8	Query Client sends a `getSubscriptionIDs` request to the server.	Confirm the element immediately nested within the `EPCISBody` element of the query document is "`GetSubscriptionIDs`".

9	Query Client sends an XML document where the element immediately nested within `EPCISBody` element is `Test`.	Confirm the response XML document from the query server is an `EPCISQueryDocument` instance where the element immediately nested within the `EPCISBody` is a "`ValidationException`".

247

248

249 2.31 Test Case Requirement 31 – Standard Business Document
250 Header in AS2 binding of Query Control Interface

Standard Business Document Header in Query Control Interface		
TPId: TCR-31		
Requirement Purpose: This Test Case Requirement confirms the inclusion of the Standard Business Document Header in the `EPCISHeader` element in the query XML sent by the client under the AS2 binding and the inclusion of a unique identifier within the Standard Business Document Header. **Requirements Tested**: M77, M78		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • Requirements for TCR-27_are satisfied • Requirements for TCR-29 are satisfied		
Step	**Step description**	**Expected results**
1	Verify that the `EPCISHeader` element is present within the `EPCISQueryDocument` element of the XML message sent by the query client.	Confirm that the "`StandardBusinessDocumentHeader`" element is present within the `EPCISHeader` element.
2	Verify that the `InstanceIdentifier` element is present within the `DocumentIdentification` element within the `StandardBusinessDocumentHeader` element.	Confirm that the value of the `InstanceIdentifier` element is unique among all messages sent from the query client to the server.

251

252

2.32 Test Case Requirement 32 – Response XML in AS2 binding of Query Control Interface

Response XML in AS2 binding of Query Control Interface		
TPId: TCR-32		
Requirement Purpose: This Test Case Requirement confirms that a response message sent by the query server under the AS2 binding of EPCIS QueryControl Interface is an EPCISQueryDocument. **Requirements Tested**: M80, M81		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • Requirements for TCR-27 are satisfied • Requirements for TCR-30 are satisfied		
Step	**Step description**	**Expected results**
2	Query Client sends a request to query server according to TCR-29	Confirm that the root element of the response XML from query server is "`EPCISQueryDocument`" with `EPCISHeader` and `EPCISBody` elements nested within.
3	Verify that response is for `getQueryNames` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`GetQueryNamesResult`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".
4	Verify that response is for `getStandardVersion` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`GetStandardVersionResult`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".
5	Verify that response is for `getVendorVersion` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`GetVendorVersionResult`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".

6	Verify that response is for `poll` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`QueryResults`" or "`QueryParameterExeption`" or "`QueryTooLargeException`" or "`QueryTooComplexException`" or "`NoSuchNameException`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".
7	Verify that response is for `subscribe` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`SubscribeResult`" or "`QueryParameterExeption`" or "`QueryTooComplexException`" or " "`NoSuchNameException`" or "`InvalidURIException`" or "`DuplicateSubscriptionException`" or "`SubscriptionControlsException`" or "`SubscribeNotPermittedException`" "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".
8	Verify that response is for `unsubscribe` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`Unsubscribe Result`" or "`NoSuchSubscriptionException`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".
9	Verify that response is for `getSubscriptionIDs` request.	Confirm the element immediately nested within the `EPCISBody` element of the response is either "`UnsubscribeResult`" or "`NoSuchNameException`" or "`SecurityException`" or "`ValidationException`" or "`ImplementationException`".

255

256

2.33 Test Case Requirement 33 – Response XML in AS2 Binding of Query Control Interface

Response XML in AS2 binding of Query Control Interface		
TPId: TCR-33		
Requirement Purpose: : This Test Case Requirement confirms the proper usage of the Standard Business Document Header in the response message sent by the query server under the AS2 binding of EPCIS Query Control Interface. **Requirements Tested**: M82-85		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup. • Data has been loaded on Repository Server. • Requirements for TCR-27 are satisfied. • Requirements for TCR-30 are satisfied.		
Step	**Step description**	**Expected results**
1	Check the response message sent by query server.	Confirm `EPCISHeader` element is present in the response.
2	Check the `EPCISHeader` element in the response.	Confirm that it contains `StandardBusinessDocumentHeader` element.
3	Check the sub elements of `StandardBusinessDocumentHeader` element.	Confirm that `BusinessScope` element is present.
3	Check the sub elements of `BusinessScope` element.	Confirm that `Scope` element is present.
4	Check the sub elements of `Scope` element.	Confirm that it contains `CorrelationInformation`, `Type` and `InstanceIdentifier` elements.
4	Check the sub elements of `CorrelationInformation` element.	Confirm that `RequestingDocumentInstanceIdentifier` element is present.
5	Check the value of `RequestingDocumentInstanceIdentifier` element.	Confirm that value is same as the value of `InstanceIdentifier` element from the Standard Business Document Header of the corresponding request.
6	Check the value of `Type` element.	Confirm that value is `EPCISQuery`.
7	Check the value of `InstanceIdentifier` element.	Confirm that value is `EPCIS`.

2.34 Test Case Requirement 34 – AS2 Binding for Query Callback Interface

AS2 Binding for Query Callback Interface		
TPId: TCR-34		
Requirement Purpose: This Test Case Requirement confirms that the AS2 destination specified in EPCIS Query Callback Interface should follow proper syntax. **Requirements Tested**: M98		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup. • Data has been loaded on Repository Server. • Requirements for TCR-27 are satisfied. • The subscription request is made and it confirms to TCR-30 conditions.		
Step	**Step description**	**Expected results**
1	Check the AS2 destination URI.	Confirm that it follows the syntax as `as2:`*`remainder-of-URI.`*
2	Check the syntax of *`remainder-of-URI.`*	Confirms that the *`remainder-of-URI`* • Identifies a specific AS2 communication profile to be used by the EPCIS Service to deliver information to the subscriber. • Has syntax in accordance to the EPCIS Service to which the subscription is made and it conforms to URI syntax as defined by [RFC2396].

2.35 Test Case Requirement 35 – AS2 Binding for Query Callback Interface

AS2 Binding for Query Callback Interface		
TPId: TCR-35		
Requirement Purpose: This Test Case Requirement confirms that the payload delivered by the query server to the query client under the AS2 binding of EPCIS Query Callback Interface should be an XML document whose root element is the `EPCISQueryDocument` which contains the `EPCISBody` element. **Requirements Tested**: M100		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • Requirements for TCR-27 are satisfied • The subscription request is made and it confirms to TCR-34 conditions.		
Step	**Step description**	**Expected results**
1	For the Query Callback Interface Method `callbackResults`.	Confirm that the `EPCISBody` has "`QueryResults`" element.
2	For the Query Callback Interface Method `callbackQueryTooLargeException`.	Confirm that the `EPCISBody` has "`QueryTooLargeException`" element.
3	If the Query Callback Interface Method is `callbackImplementationException`.	Confirm that the `EPCISBody` has "`ImplementationException`" element.
4	Verify that the `QueryResults` or the `QueryTooLargeException` or the `ImplementationException` element contains the `queryName element`.	Confirm that `queryName` is same as the one supplied in call to `subscribe`.
5	Verify that the `QueryResults` or the `QueryTooLargeException` or the `ImplementationException` element contains the `subscriptionID element`.	Confirm that "`subscriptionID`" is same as the one supplied in call to `subscribe`.

2.36 Test Case Requirement 36 – Query Server Response through Query Callback Interface

Query Server Response through Query Callback Interface		
TPId: TCR-36		
Requirement Purpose: This Test Case Requirement confirms that the payload to be delivered by the query server to the query client through EPCIS Query Callback Interface is an XML document whose root element is the `EPCISQueryDocument` containing the `EPCISBody` element. **Requirements Tested**: M88		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been setup • Data has been loaded on Repository Server • The subscription request has been made.		
Step	**Step description**	**Expected results**
1	For the Query Callback Interface Method `callbackResults`.	Confirm that the `EPCISBody` has "`QueryResults`" element.
2	For the Query Callback Interface Method `callbackQueryTooLargeException.`	Confirm that the `EPCISBody` has "`QueryTooLargeException`" element.
3	If the Query Callback Interface Method is `callbackImplementationException.`	Confirm that the `EPCISBody` has "`ImplementationException`" element.
4	Verify that the `QueryResults` or the `QueryTooLargeException` or the `ImplementationException` element contains the `queryName element.`	Confirm that `queryName` is same as the one supplied in call the to `subscribe.`
5	Verify that the `QueryResults` or the `QueryTooLargeException` or the `ImplementationException` element contains the `subscriptionID element.`	Confirm that "`subscriptionID`" is same as the one supplied in the call to `subscribe.`

2.37 Test Case Requirement 37 – Interpretation of the Response Code on the Query Callback

Interpretation of the Response Code on the Query Callback		
TPId: TCR-37		
Requirement Purpose: This Test Case Requirement confirms that the Query Client is properly interpreting the response code received from the Query Server. **Requirements Tested**: M93, M97		
Pre-test conditions: • EPCIS Accessing App and EPCIS Repository servers have been set up • Data has been loaded on Repository Server • HTTP or HTTPS Binding has been established • The subscription request has been made.		
Step	**Step description**	**Expected results**
1	When the Query Client receives a response code of 200-299 through the Query Callback Interface for an HTTP Binding.	Confirm that the Query Client is treating this response code as a valid value without generating an error.
2	When the Query Client receives a response code of 200-299 through the Query Callback Interface for an HTTPS Binding.	Confirm that the Query Client is treating this response code as a valid value without generating an error.

2.38 Test Case Requirement 38 – HTTPS Binding Configuration

HTTPS Binding Configuration for Query Callback		
TPId: TCR-38		
Requirement Purpose: This Test Case Requirement confirms that the HTTPS Binding has been configured for the Query Callback Interface. **Requirements Tested**: M96		
Pre-test conditions:		
Step	**Step description**	**Expected results**
1	For HTTPS Binding on the Query Callback Interface	Confirm that HTTP over TLS conforming to RFC2818 has been used. Confirm that TLS as defined in RFC2246 has been implemented. Confirm that for TLS, at least TLS_RSA_WITH_AES_128_CBC_SHA cipher suite as defined in RFC3268 with CompressionMethod.null is supported.

2.39 Test Case Requirement 39 – Provide and capture events that correctly represent the value of the eventTimeZoneOffset field

Provide and capture events that correctly represent the value of eventTimeZoneOffset		
TPId: TCR-39		
Requirement Purpose: This Test Case Requirement confirms that an implementation can provide and capture events that correctly represent the value of the eventTimeZoneOffset field **Requirements Tested**: M7		
Pre-test conditions: • Create a set of events that contains all four event types with data in each standard field, multiple extension fields, and the eventTimeZoneOffset field. • Otherwise the EPCIS system should not contain any EPCIS events		
Step	**Step description**	**Expected results**
1	Capture client provides events with a correct value of eventTimeZoneOffset to a capture server	Confirm that the capture server captures the events
2	Capture client provides events with an incorrect value of eventTimeZoneOffset to a capture server	Confirm that the capture server does not capture the events

2.40 Test Case Requirement 40 – Query Interface Exceptions

Query Interface Exceptions		
TPId: TCR-40		
Requirement Purpose: **Requirements Tested**: M20, M24-28, M30, M32		
Pre-test conditions: • The EPCIS Query server is running. • An implementation supported binding is selected: XML over AS2 or SOAP over HTTP (WSDL)		
Step	**Step description**	**Expected results**
1	Invoke the EPCIS Query Control Interface subscribe method with an incorrect "dest" argument (a string that does not conform to a valid URI). All other subscriber parameters must be valid. (M17)	Verify the Query Server implementation raises an InvalidURIException
2	Invoke the EPCIS Query Control Interface subscribe method with an empty dest argument and no pre-arranged URI. All other subscriber parameters must be valid. (M17)	Verify the Query Server implementation raises an InvalidURIException
3	Invoke the EPCIS Query Control Interface subscribe method when a parameter required by a query is omitted or is supplied with an empty value (M21)	Verify the Query Server implementation raises a QueryParameterException
4	Invoke the EPCIS Query Control Interface poll method when a parameter required by a query is omitted or is supplied with an empty value (M21)	Verify the Query Server implementation raises a QueryParameterException
5	Invoke the EPCIS Query Control Interface poll method when a parameter is supplied whose name does not correspond to any parameter name defined by the query (M22)	Verify the Query Server implementation raises a QueryParameterException
6	Invoke the EPCIS Query Control Interface subscribe method when a parameter is supplied whose name does not correspond to any parameter name defined by the query (M22)	Verify the Query Server implementation raises a QueryParameterException

7	Invoke the EPCIS Query Control Interface poll method when two parameters are supplied with the same name (M23)	Verify the Query Server implementation raises a QueryParameterException
8	Invoke the EPCIS Query Control Interface subscribe method when two parameters are supplied with the same name (M23)	Verify the Query Server implementation raises a QueryParameterException
9	Invoke the EPCIS Query Control Interface poll method when any other constraint imposed by the query is violated – e.g.: range of permitted values for a parameter parameters are supplied with the same name (M24)	Verify the Query Server implementation raises a QueryParameterException
10	Invoke the EPCIS Query Control Interface subscribe method when any other constraint imposed by the query is violated – e.g.: range of permitted values for a parameter parameters are supplied with the same name (M24)	Verify the Query Server implementation raises a QueryParameterException
11	Invoke the EPCIS Query Control Interface subscribe method with both the “schedule” and “trigger” specified in the SubscriptionControls parameter (M25)	Verify the Query Server implementation raises a SubscriptionControlsException
12	Invoke the EPCIS Query Control Interface subscribe method with neither the “schedule” and “trigger” specified in the SubscriptionControls parameter (M25)	Verify the Query Server implementation raises a SubscriptionControlsException
13	Invoke the EPCIS Query Control Interface subscribe method with a SubscriptionControls query schedule field that does not conform to the grammar in Section 8.2.5.3 of the EPCIS specification (M27).	Verify the Query Server implementation raises a SubscriptionControlsException
14	Invoke the EPCIS Query Control Interface subscribe method when the EPCIS Query server knows that it cannot honor a QuerySchedule without deviating widely from the request. (M29) **Note**: The amount of time deviation allowed before a QuerySchedule cannot be honored and an exception is raised may vary among implementations. The tester should be able to adjust the QuerySchedule timing to a value that will cause the specific implementation under test to raise the SubscriptionControlsException.	Verify the Query Server implementation raises a SubscriptionControlsException

15	Repeat steps 1 and 14 for the other binding (if supported by the implementation)	

288

289

2.41 Test Case Requirement 41 – Basic Subscription Operation Verification

Basic Subscription Operation Verification

TPId: TCR-41

Requirement Purpose: The purpose of this test is to verify the basic Query operation for the subscription service.

Requirements Tested: M23, M29, M31, M34

Pre-test conditions:

- The implementation's EPCIS Repository has appropriately populated EPCIS ObjectEvents so valid results, including ObjectEvents, are returned in the QueryResult.
- The system under test is "lightly loaded"
- An implementation supported binding is selected: XML over AS2 or SOAP over HTTP (WSDL)

Subscribe Parameters

QueryName			
"SimpleEventQuery"			
QueryParams			
Name	**Value**		
"eventType"	"ObjectEvent"		
"GE_eventTime"	<empty>		
"dest"			
<Valid Destination URI>			
SubscriptionID			
<Valid String>			
SubscriptionControls			
QuerySchedule	**Trigger**	**InitialRecordTime**	**ReportIfEmpty**
A string in accordance with the requirements of section 8.2.5.3 of the EPCIS specification. The schedule period should be set to greater than 2 minutes so verification of the 2 response within 2 minutes requirement is tested.	unspecified	Set earlier than recordTime of EPCISEvents in repository	True

Step	**Step description**	**Expected results**

1	Invoke the subscribe method with valid parameters as specified in the pre-test conditions. Wait for the first query to be executed according to the query schedule specified.	A periodic subscription result should be returned based on the setting of the QuerySchedule parameter. ObjectEvents should be returned in the first QueryResults sent to the dest. The "GE_eventTime" should not have an effect on the ObjectEvents returned. All QueryResults should be received within two minutes of their scheduled time. The QueryName should be in the QueryResult. The SubscriptionID should be in the QueryResult. The first time the query is executed for the subscription, the only events considered are those whose recordTime field is greater than or equal to initialRecordTime specified when the subscription was created.
2	Allow the subscription to remain active. Do not add new EPCIS events to the EPCIS repository. Allow enough time for new query to be executed according to the schedule set in the subscribe method.	For each execution of the query following the first, the only events considered are those whose recordTime field is greater than or equal to the time when the query was last executed. Since no new EPCIS event were added since the previous query, a query result with no events in it should be sent since "report if empty" is true.
3	Add new EPCIS events to the EPCIS repository before the next query result is to be returned. Wait for the query to be executed according to the schedule set in the subscribe method.	A query result should be returned containing only the new EPCIS events that were added.
4	Invoke the unsubscribe method to end the test.	
5	Repeat steps 1 to 4 for the other binding (if supported by the implementation)	

292

293

2.42 Test Case Requirement 42 – Basic Poll Operation Verification

Basic Poll Operation Verification

TPId: TCR-42

Requirement Purpose: The purpose of this test is to verify the basic Query operation for the poll service.

Requirements Tested: M23, M33, M35

Pre-test conditions:

- The implementation's EPCIS Repository has appropriately populated EPCIS ObjectEvents so valid results, including ObjectEvents, are be returned in the QueryResult.
- The system under test is "lightly loaded"
- An implementation supported binding is selected: XML over AS2 or SOAP over HTTP (WSDL)

Poll Parameters

QueryName	
"SimpleEventQuery"	
QueryParams	
Name	**Value**
"eventType"	"ObjectEvent"

Step	Step description	Expected results
1	Invoke the poll method with valid poll method parameters as specified in the pre-test conditions.	The QueryName should be in the QueryResult. The SubscriptionID should NOT be in the QueryResult.
2	Repeat steps 1 for the other binding (if supported by the implementation)	

299 **2.43**

300 **2.43**

301

2.44 Test Case Requirement 44 – Capture Server Message Queue

Capture Server Message Queue		
TPId: TCR-44		
Requirement Purpose: Confirms that capture events can be sent to the capture server via message queue. **Requirements Tested**: M58		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create a capture event consisting of an EPCISDocument having a single EPCISEvent. Send it to the capture server using the server's message queue mechanism.	Server stores the event's data.

2.45 Test Case Requirement 45 – Capture Server HTTP Input

Capture Server HTTP Input		
TPId: TCR-45		
Requirement Purpose: Confirms that capture events can be sent to the capture server via HTTP POST. **Requirements Tested**: M61-62		
Pre-test conditions: none		
Step	**Step description**	**Expected results**
1	Create a capture event consisting of an EPCISDocument having a single EPCISEvent. Send it to the capture server using HTTP POST to the server's specified URL.	Provided the HTTP return status code is 200, the event's data must be stored. When the HTTP return status code is in the range 300 through 599 inclusive, then none of the event's data must be stored.

2.46 Test Case Requirement 47 – Class Level Identified Objects & Objects & Aggregation/Disaggregation

Class Level Identified Objects & Aggregation/Disaggregation		
TPId: TCR-47		
Requirement Purpose: This Test Case Requirement confirms that the capture interface will accept aggregation/disaggregation events with new class level identified objects quantity list. **Requirements Tested**: M110, M111, M114		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema.
• There are one or more events in the document
• Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create documents with a single instance level identified parent and one or more class level identified child quantity lists (AggregationEvent with action ADD and AggregationEvent with action DELETE for disaggregation)	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 XML Schema.
2	Create one or more events with child class level objects using one GTIN identifiers (as an EPC pattern URI)	• Document complies with epcis 1.1 schema. Identifiers conform to the TDS and they appear in the proper place it the aggregation event. •
3	Create one or more events with child class level objects using 2 or more different GTIN identifiers (as an EPC pattern URI)	•
4	Create one or more events with child class level objects using GTIN plus LOT identifiers (as an LGTIN URI)	• Document complies with epcis 1.1 schema. Identifiers conform to the TDS and they appear in the proper place itn he aggregation event. •

5	Create one or more events with child class level objects using different LOT identifiers of the same GTIN identifier (as an LGTIN URI)	• Identifier conforms to TDS • Identifiers appear in the right place of the Aggregation event
	Fields to be tested: `childQuantityList` Actions to be tested: `ADD, DELETE`	

311

312

313 2.48 Test Case Requirement 48 – Transformation Event

Transformation Event		
TPId: TCR-48		
Requirement Purpose: This Test Case Requirement confirms the capture interface will accept transformation events using both instance and class level identifiers. **Requirements Tested**: M116, M117		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema
• There are one or more events in the document
• Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create transformation event document for instance-level input and instance-level output with transformation ID	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
2	Create transformation event document for instance-level input and class-level output	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
3	Create transformation event document for class-level input and instance-level output	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.

4	Create transformation event document for class-level input and class-level output	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
5		
	Fields to be tested: `inputEPCList , inputQuantityList , outputEPCList , outputQuantityList , transformationID`	

314

315

2.49 Test Case Requirement 49 – Class Quantity Identification

Class Quantity Identification		
TPId: TCR-49		
Requirement Purpose: This Test Case Requirement confirms that the capture interface will accept class level identifiers with quantities in each event type. **Requirements Tested**: M104 – M109		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema.
• There are one or more events in the document
• Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create one document for each event type (ObjectEvent, AggregationEvent, TransactionEvent, and TransformationEvent)	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
2	In each event create one or more instances of the quantity element	Verify that the structure is valid and contains data elements for: • epcClass • quantity • uom
3	For each instance of epcClass in each quantity element created	• Verify passes all general URI tests

2.50 Test Case Requirement 50 – Source/Destination

TCR5: Identification of Parties and Locations		
TPId: TCR-50		
Requirement Purpose: This Test Case Requirement confirms that the capture interface will accept source and destination identifiers in each event type. **Requirements Tested**:		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema.
• There are one or more events in the document
• Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create one or more documents for each event type (ObjectEvent, AggregationEvent, TransactionEvent, and TransformationEvent)	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
2	In each event create one or more instances of the sourceList elements	Verify that the structure is valid and contains data elements for: • source • type attribute
3	For each instance of the type attribute in each sourceList element created	• Verify passes all general URI tests
4	In each event create one or more instances of the destinationList elements	Verify that the structure is valid and contains data elements for: • destination • type attribute
5	For each instance of the type attribute in each sourceList element created	• Verify passes all general URI tests
	Fields to be tested: `sourceList, destinationList` Business transaction types to be tested: `owning_party, possessing_party, location`	

2.51 Test Case Requirement 51 – Instance/Lot Master Data (ILMD)

Instance/Lot Master Data (ILMD)		
TPId: TCR-51		
Requirement Purpose: This Test Case Requirement confirms that the capture interface will accept user extensions made available to record instance/lot master data in Object and Transformation event types. **Requirements Tested**: M113		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema.
• There are one or more events in the document
• Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create one or more documents for each event type (ObjectEvent with action ADD and TransformationEvent with one or more outputEPCList or outputQuantityList elements)	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
2	In each event define two or more custom ILMD fields within the user extension of the ilmd data element	• Verify that the custom data elements within the ILMD extension are well-formed. • Verify that the document including the ILMD extension validates against the EPCIS 1.1 schema • Verify that the custom ILMD data element comply with the EPCIS 1.1 requirements
3	An `ObjectEvent` SHALL NOT contain `ilmd` if `action` is `OBSERVE` or `DELETE`.	ObjectEvent with actin\on OBSERVE or DELETE that contains a ilmd section must cause an error.
	Fields to be tested: `ilmd`	

2.52 Test Case Requirement 52 – Queries

Queries		
TPId: TCR-52		
Requirement Purpose: This Test Case Requirement confirms the query interface is successfully executed using the newly defined parameters in v1.2. **Requirements Tested**: M19, M37, M38, M118, M119		

Pre-test conditions:
• Document passes xml validation
  ○ XML is well formed.
  ○ XML is valid according to EPCIS 1.1 schema.
• There are one or more events in the document
  ○ Provide the document-to-test via file upload or via HTTP POST.		

Step	**Step description**	**Expected results**
1	Create one or more documents for each new query parameter.	Verify that in the submitted document: • Document is a capture-acceptable form (EPCISDocument or EPCISQueryResponse meeting requirements) • Document complies with the EPCIS 1.1 Requirements.
2	EQ_source_type : This is not a single parameter, but a family of parameters. If a parameter of this form is specified, the result will only include events that (a) include a sourceList; (b) where the source list includes an entry whose type subfield is equal to type extracted from the name of this parameter; and (c) where the source subfield of that entry is equal to one of the values specified in this parameter.	Confirm that the only events with a matching *source* with the correct *type* are returned.
3	EQ_destination_Type : This is not a single parameter, but a family of parameters. If a parameter of this form is specified, the result will only include events that (a) include a destinationList; (b) where the destination list includes an entry whose type subfield is equal to type extracted from the name of this parameter; and (c) where the destination subfield of that entry is equal to one of the values specified in this parameter.	Confirm that the only events with a matching *destination* with the correct *type* are returned.

4	EQ_transformationID : If this parameter is specified, the result will only include events that (a) have a transformationID field (that is, TransformationEvents or extension event type that extend TransformationEvent); and where (b) the transformationID field is equal to one of the values specified in this parameter.	Confirm that the only events with a matching *transformationID* are returned.
5	MATCH_inputEPC : If this parameter is specified, the result will only include events that (a) have an inputEPCList (that is, TransformationEvent or an extension event type that extends TransformationEvent); and where (b) one of the EPCs listed in the inputEPCList field matches one of the EPC patterns or URIs specified in this parameter. If this parameter is omitted, events are included regardless of their inputEPCList field or whether the inputEPCList field exists.	Confirm that the only events with a matching *inputEPC* are returned.
6	MATCH_outputEPC : If this parameter is specified, the result will only include events that (a) have an outputEPCList (that is, TransformationEvent or an extension event type that extends TransformationEvent); and where (b) one of the EPCs listed in the outputEPCList field matches one of the EPC patterns or URIs specified in this parameter. If this parameter is omitted, events are included regardless of their outputEPCList field or whether the outputEPCList field exists.	Confirm that the only events with a matching *outputEPC* are returned.
7	MATCH_inputEPCClass : If this parameter is specified, the result will only include events that (a) have an inputQuantityList field (that is, TransformationEvent or extension event types that extend it); and where (b) one of the EPC classes listed in the inputQuantityList field (depending on event type) matches one of the EPC patterns or URIs specified in this parameter.	Confirm that the only events with a matching *inputEPCClass* are returned.

8	MATCH_outputEPCClass : If this parameter is specified, the result will only include events that (a) have an outputQuantityList field (that is, TransformationEvent or extension event types that extend it); and where (b) one of the EPC classes listed in the outputQuantityList field (depending on event type) matches one of the EPC patterns or URIs specified in this parameter.	Confirm that the only events with a matching *outputEPCClass* are returned.
9	MATCH_epcClass : If this parameter is specified, the result will only include events that (a) have a quantityList or a childQuantityList field (that is, ObjectEvent, AggregationEvent, TransactionEvent or extension event types that extend one of those three); and where (b) one of the EPC classes listed in the quantityList or childQuantityList field (depending on event type) matches one of the EPC patterns or URIs specified in this parameter. The result will also include QuantityEvents whose epcClass field matches one of the EPC patterns or URIs specified in this parameter.	Confirm that the only events with a matching *epcClass* are returned.
10	MATCH_anyEPCClass : If this parameter is specified, the result will only include events that (a) have a quantityList, childQuantityList, inputQuantityList, or outputQuantityList field (that is, ObjectEvent, AggregationEvent, TransactionEvent, TransformationEvent, or extension event types that extend one of those four); and where (b) one of the EPC classes listed in any of those fields matches one of the EPC patterns or URIs specified in this parameter. The result will also include QuantityEvents whose epcClass field matches one of the EPC patterns or URIs specified in this parameter.	Confirm that the only events with a matching *anyEPCClass* are returned.
11	EQ_ILMD_fieldname - string : Analogous to EQ_fieldname, but matches events whose ILMD area contains a field having the specified fieldname whose value matches one of the specified values.	Confirm that the only events with a matching *fieldname* with the type *string* are returned.

12	EQ_ILMD_fieldname : Analogous to EQ_fieldname, GT_fieldname, GE_fieldname, GE_fieldname, LT_fieldname, and LE_fieldname, respectively, but matches events whose ILMD area contains a field having the specified fieldname whose integer, float, or time value matches the specified value according to the specified relational operator.	Confirm that the only events with a matching *fieldname* are returned.
13	GT_ILMD_fieldname : Analogous to EQ_fieldname, GT_fieldname, GE_fieldname, GE_fieldname, LT_fieldname, and LE_fieldname, respectively, but matches events whose ILMD area contains a field having the specified fieldname whose integer, float, or time value matches the specified value according to the specified relational operator.	Confirm that the only events of result with *fieldname* value greater than specified are returned.
14	GE_ILMD_fieldname : Analogous to EQ_fieldname, GT_fieldname, GE_fieldname, GE_fieldname, LT_fieldname, and LE_fieldname, respectively, but matches events whose ILMD area contains a field having the specified fieldname whose integer, float, or time value matches the specified value according to the specified relational operator.	Confirm that the only events of result with *fieldname* value greater or equal than specified are returned.
15	LT_ILMD_fieldname : Analogous to EQ_fieldname, GT_fieldname, GE_fieldname, GE_fieldname, LT_fieldname, and LE_fieldname, respectively, but matches events whose ILMD area contains a field having the specified fieldname whose integer, float, or time value matches the specified value according to the specified relational operator.	Confirm that the only events of result with *fieldname* value lower than specified are returned.

16	LE_ILMD_fieldname : Analogous to EQ_fieldname, GT_fieldname, GE_fieldname, GE_fieldname, LT_fieldname, and LE_fieldname, respectively, but matches events whose ILMD area contains a field having the specified fieldname whose integer, float, or time value matches the specified value according to the specified relational operator.	Confirm that the only events of result with *fieldname* value lower or equal than specified are returned.
17	EXISTS_ILMD_fieldname : Like EXISTS_fieldname as described above, but events that have a non-empty field named fieldname in the ILMD area. Fieldname is constructed as for EQ_ILMD_fieldname. Note that the value for this query parameter is ignored.	Confirm that the only events with *fieldname* field are returned.
	parameters to be tested: `EQ_source_type, EQ_destination_Type, EQ_transformationID, MATCH_inputEPC, MATCH_outputEPC, MATCH_inputEPCClass, MATCH_outputEPCClass, MATCH_epcClass, MATCH_anyEPCClass, EQ_ILMD_fieldname, EQ_ILMD_fieldname, GT_ILMD_fieldname, GE_ILMD_fieldname, LT_ILMD_fieldname, LE_ILMD_fieldname, EXISTS_ILMD_fieldname`	

327

328

3 References

[W3C-Conformance] D. Dardailler, "Conformance Testing and Certification Model for W3C Specifications," W3C Note, http://www.w3.org/QA/2002/01/Note-qa-certif-20020102.html, January 2002.

K. Traub, editor, "EPC Information Services (EPCIS) Version 1.1 Specification, http://www.gs1.org/gsmp/kc/epcglobal/epcis/epcis_1_1-standard-20140520.pdf, May 2014

