

The B2B Plugtests™ event
7 to 11 July 2008
At ETSI, Sophia Antipolis
France

Technical Overview and
Organization
(V0.5)

- 1 Introduction..... 3
 - 1.1 The ETSI Plugtests – a professional service 3
 - 1.2 The B2B Plugtests event..... 3
- 2 Test Program..... 4
 - 2.1 Testing objectives..... 4
 - 2.2 Testing Methods..... 5
 - 2.3 Organization and Participants 7
 - 2.4 General Test Scenarios..... 10
 - 2.4.1 Conformance Test scenario (Standards varying) 10
 - 2.4.2 Interoperability Test scenarios (should be investigated) 10
 - 2.5 List of Test Cases..... 12

1 Introduction

1.1 The ETSI Plugtests – a professional service

The ETSI Plugtests service organizes cost neutral Inter-operability events, which are open to all organizations (not only ETSI members) and several domains (not only Telecoms)

Plugtests are addressed to any company developing a product such as operators, vendors or equipment manufacturers, content providers or application providers

Goals:

- Verify Inter-operability
- Debug
- Understand – learn – consolidate technical vision
- Feedbacks to Standardization

1.2 The B2B Plugtests event

eInvoicing, eProcurement, and eBusiness user communities are currently investigating what their future eBusiness infrastructure should be. Many small companies will be required to use electronic documents and handle the business relationships with their suppliers, customers and even with administrations over EDI solutions.

Decisions in this area are not just based on technology and standards – they require a level of confidence that these technologies can solve the specific challenges the users have. Ensuring the Interoperability between the different solutions and standards is the key for the global success of EDI.

In this prospect ETSI's Plugtests™ service in cooperation with the CEN/ISSS eBES group is organizing this interoperability test event. In addition, together with our partners **KIEC/KorBIT** and **NIST** there will be the opportunity to perform conformance testing for B2B communication: ebXML, W/S, AS2, FTP, etc. as well as business document mapping and business processes.

This event will provide the eInvoicing, eProcurement, and eBusiness user communities the opportunity to:

- Apply B2B communication, messaging technologies and tools to their own business documents and transactions.
- Test the interoperability with other partners in their own application space.
- Test the interoperability between different messaging standards using Gateways.
- Refine the design of an interoperable solution with their partners and set the basis for interoperability rules and best practices within their community.

2 Test Program

NOTE: this document gives only an overview of the test program. More detailed test suites involved are available in adjunct documents.

2.1 Testing objectives

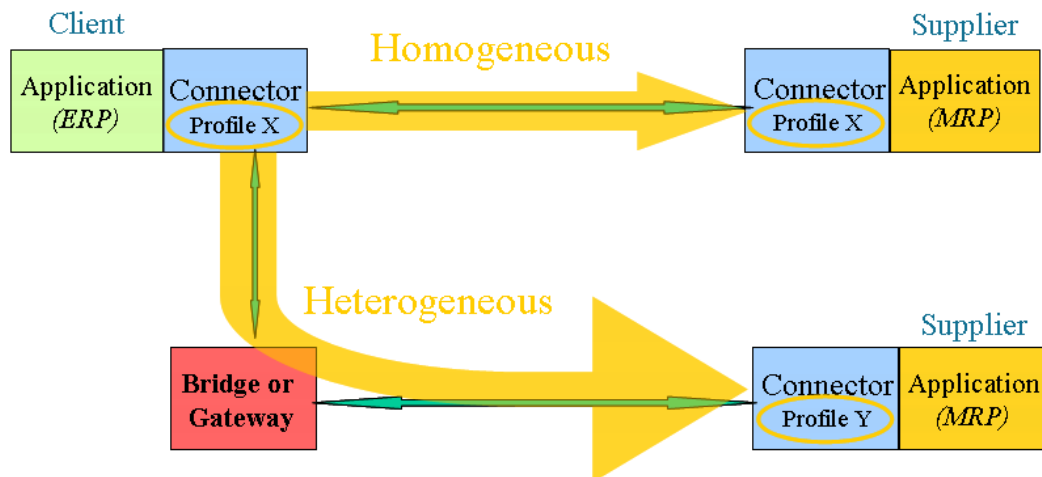
Standards Involved:

The B2B Plugtests event will focus on the following technologies and standards which will be the primary targets:

- Infrastructure, Messaging and Transaction Protocols and Standards
 - Simple Object Access Protocol Version 1.0, 1.1,
 - ebXML messaging Specification Version 2, 3,
 - WS-I Basic Profile Version 1.0, 1.1, 1.2,
 - WS-I Attachments Profile Version 1.0,
 - WS-I Basic Security Profile Version 1.0,
 - WS-I Reliable Secure Profile Version 1.0,
 - IBM Reliable Asynchronous Messaging Protocol (RAMP) Version 1.0,
 - Action Script Version 2.0
- Business Object Document Profiles and Standards
 - Open Business Objects for EDI (OBOE)
 - OAGIS 8.0, 9.0, 9.1, 9.2
 - OAGIS 9.0 Naming and Design Rules
 - SAP Global Data Type
 - KIEC Schema Profile
 - UN/CEFACT – Core Components Technical Specification (CCTS)
- Business Process, Discovery and Collaboration Profiles
 - Integration with services on the back-end, if applicable (e.g. Web services that wrap applications modules), or with back-end queuing (JMS),
 - ebXML ebBP 2.0 and UML and WSDL for business transactions modelling and scripting, possibly BPM for processes, CPPA for agreement representation, Registry/repository and UDDI for assets management, resource sharing,

B2B Models under Test:

The general model of B2B systems under tests is as illustrated below.



The B2B endpoints illustrated in the above figure are composed of applications (MRP, ERP) combined with “connectors”. A connector is a binding components between the application business process and the standard-based B2B process. The connector is also the component responsible for implementing standard-conforming messaging functions. Application components are responsible for producing and processing conforming business documents.

Bridges across messaging transports will also be available either as part of the interoperability environment, or as systems under tests (SUT) themselves.

Two types of communication between connector are expected:

- Homogeneous communication, when both endpoints in a B2B exchange use the same standard (messaging, documents)
- Heterogeneous communication, when both endpoints in a B2B exchange use different standards (messaging, or documents) for which the mediation of a bridge is required.

In-production deployments could be more complex, involving for instance tier 2, tier 3 suppliers. The above figure is intending to define the basic test framework.

2.2 Testing Methods

The focus is on Interoperability of an end-to-end B2B chain, including messaging, quality of service (security, reliability), document exchange and application binding. A couple of most common business transaction patterns (from UN/CEFACT) are used.

The expected test process is for groups of participants to first do conformance testing (to the standards they elected to interoperate with), then to test interoperability using the above transaction patterns.

- **Conformance testing** is using executable conformance test suite, based on formal test specifications, to test the compliance of implementation to the standards. This enables to find out lacks of compliance of the implementation functions, which can leads to interoperability issues. Conformance testing requires using specific test systems, which runs test cases against real implementations, as per the test specifications.

- **Interoperability testing** consists simply of testing that two real implementations interoperate according to the standards. It does not require specific test systems, but only interoperability test descriptions.

Both Conformance and Interoperability testing will be used during the Plugtests event. Conformance test suites and testbeds have been developed for:

- Simple Object Access Protocol Version 1.0, 1.1,
- ebXML messaging Specification Version 2, 3,
- WS-I Basic Profile Version 1.0, 1.1, 1.2,
- WS-I Attachments Profile Version 1.0,
- WS-I Basic Security Profile Version 1.0,
- WS-I Reliable Secure Profile Version 1.0,
- IBM Reliable Asynchronous Messaging Protocol (RAMP) Version 1.0,
- Action Script Version 2.0
- Open Business Objects for EDI (OBOE)
- OAGIS 8.0, 9.0, 9.1, 9.2
- OAGIS 9.0 Naming and Design Rules
- SAP Global Data Type
- KIEC Schema Profile
- UN/CEFACT – Core Components Technical Specification (CCTS)

Considering time lack and participants we will provide test services for:

- ebXML messaging Specification Version 2
- IBM Reliable Asynchronous Messaging Protocol (RAMP) Version 1.0 (Partially),
- Naming and Design Rules (NDR) Version 2.0
- KIEC Schema Profile
- UN/CEFACT – Core Components Technical Specification (CCTS)

Conformance testing can be done in two ways:

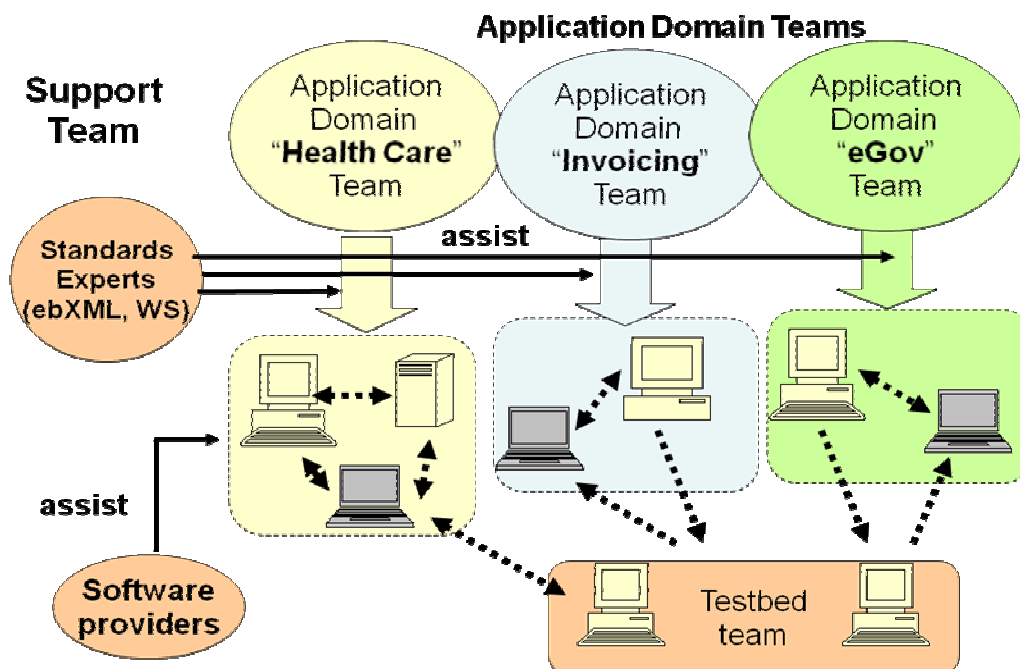
- Pre-validation tests, to enable participants to check their implementation prior to the event, using test remotely test scripts provided by one of our Plugtests partner: KIEC (Korbit),
- Conformance test sessions during the Plugtests event with the support of two Plugtests partners:
 - KIEC (Korbit)

- NIST

Interoperability Test Plans will be selected from the “Interoperability Test Base Plugtest 2008” test suite. Interoperability testing can be done below steps:

- Investigate what the Interoperability Profiles (scenarios, rules, regulations) should be tested for selected SUTs,
- Check that SUTs passed for required Conformance Testing,
- Compose the Interoperability Test Cases
- Test and debug the SUTs
- Make a report

2.3 Organization and Participants



Participants are expected to represent several industries or application domains. Test of the same standard technologies are applicable through all domains. But test sessions are mostly applicable within one application domain.

Participants will be of the following categories:

- Application domain experts (Industry, government, Health, Retail, ..),
- Product provider (commercial or Open Source),
- Test Bed experts,
- B2B experts (standardization bodies)

Two kinds of test sessions will take place:

- Conformance test sessions

- Before attending the event, participant will need to run some basic conformance test sessions to ensure a minimum level of interoperability.
- During the Plugtests, participant will have the opportunity to run conformance tests with the Test tool vendor test beds (Korbit, NIST)
- IOP (Inter-Operability) test sessions
 - The setup of IOP sessions is mainly depending on participants (availability of profiles) attending the event
 - According to the availability of Profiles 2 type of IOP sessions will be scheduled:
 - Homogeneous sessions: Connector-Connector
 - Heterogeneous sessions: Connector-Bridge-Connector

A preliminary duration for the test session duration can be evaluated in order to figure out how many interoperability session and conformance test sessions the participants are expected to execute.

Accordingly to the complexity of the settings and the necessary preliminary phases to set up between participants (Collaborating profiles,), a test session of one day looks reasonable. With a shorter duration for the test session, the time required for the test set up will represent the main part of the overall test session duration.

Of course this preliminary duration can be increased or reduced according to the attendees and the test program offered.

Over 5 days of event, we will have 4 effective days of test, so that the participants will be able to attend 1 conformance test session and 3 IOP test sessions.

| |
|--|
| <p>Preliminary Conformance Testing – Prior to the event</p> <p>Conformance Testing (Monday ~ Wednesday: 3 days)</p> <ul style="list-style-type: none"> ● Basic Transport (Monday) <ul style="list-style-type: none"> ○ SOAP ○ HTTP, HTTPs, FTP ● Messaging Standards Conformance (Monday ~ Wednesday) <ul style="list-style-type: none"> ○ ebMS 2.0 ○ Web Service (RAMP) ● Document Profile Conformance (Monday ~ Wednesday) <ul style="list-style-type: none"> ○ NDR 2.0 ○ CCTS ○ KIEC Profile |
| <p>Interoperability Testing (Tuesday ~ Thursday: 3 days)</p> <ul style="list-style-type: none"> ● Investigating IOP requirements (Tuesday ~ Wednesday) <ul style="list-style-type: none"> ○ Homogeneous Test Scenarios ○ Heterogeneous Test Scenarios |

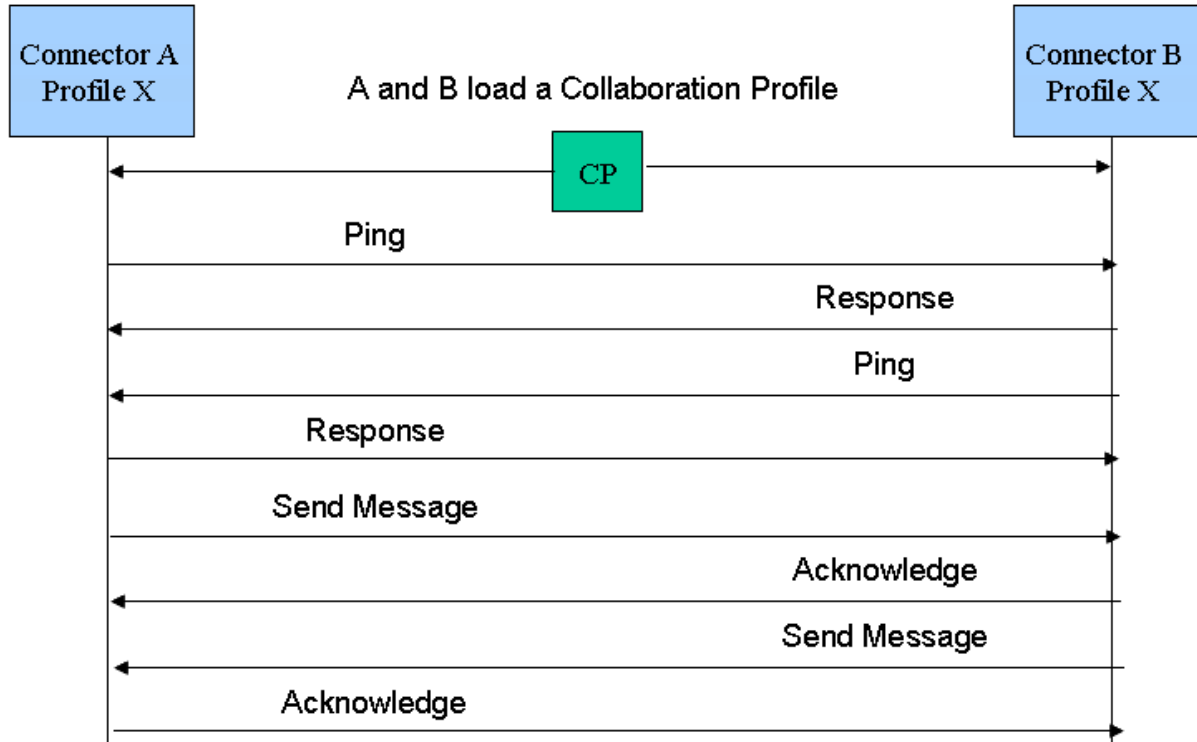
- Test Execution (Thursday)
 - ebXML
 - Web Service
 - EDIFACT
 - Mixed Model (Heterogeneous)

Discussion and Documentation (Friday)

2.4 General Test Scenarios

Generic test scenarios will be provided as guidance for the participants to proceed with their interoperability test sessions. These test scenarios are divided into 2 groups, according to two test types: Conformance and Interoperability.

2.4.1 Conformance Test scenario (Standards varying)

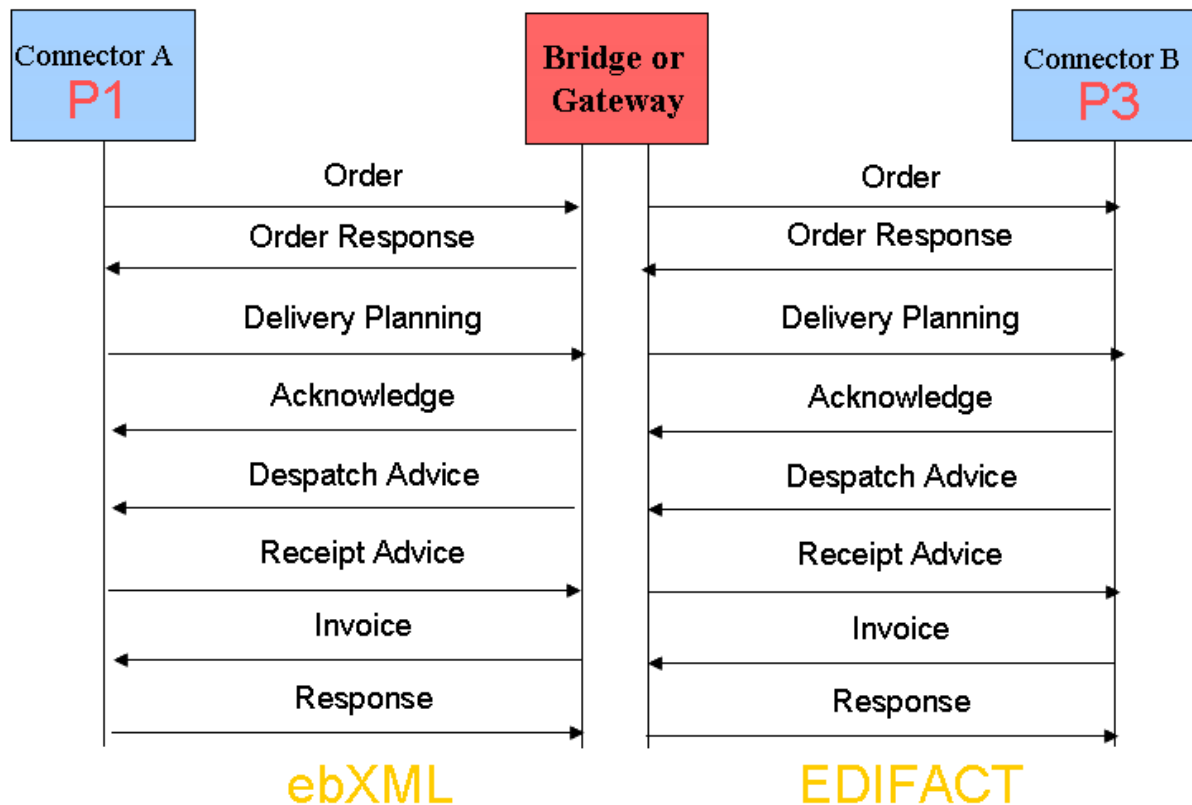


2.4.2 Interoperability Test scenarios (should be investigated)

Connector – connector



Connector – Bridge – connector



2.5 List of Test Cases

ebMS 2.0 Test Cases

| Test Case ID | Test Requirements |
|---------------|---|
| testcase_2_1 | SOAP message must be in root part of MIME message admin |
| testcase_2_2 | All MIME parts must have a CID or Content-Location admin |
| testcase_2_3 | Message Package Content-Type contains a type attribute of 'text/xml' admin |
| testcase_2_4 | Message Package Content-Type has the value of 'multipart/related' in the multipart/related message admin |
| testcase_2_5 | Message Package Content-Type has the value of 'text/xml' in the plain message admin |
| testcase_2_6 | The MIME header in Message Package contains boundary attribute admin |
| testcase_2_7 | The MIME header in Message Package contains start attribute identifying the MIME part with the ebXML Envelope admin |
| testcase_2_8 | The plain SOAP message with no payload must be processed admin |
| testcase_2_9 | The multipart message with no payload must be processed admin |
| testcase_2_10 | Soap message package Content-Type is 'text/xml' admin |
| testcase_2_11 | Content-Type header contains a 'charset' attribute specifying character set used admin |
| testcase_2_12 | If the encoding declaration exists in the SOAP message, it SHALL be equivalent to charset attribute admin |
| testcase_2_13 | UTF-8 is the xml encoding value of the SOAP message admin |
| testcase_2_14 | If the Message Package contains an application payload, it should be enclosed in Payload Container admin |
| testcase_2_15 | If there is no payload within the Message Package then a Payload Container must not be present admin |
| testcase_2_16 | The contents of each Payload Container must be a matching payload for each manifest reference admin |
| testcase_2_17 | Any message with unrecognized MIME headers must be processed admin |
| testcase_2_18 | If the XML declaration exists in the SOAP message, it must contain Version number admin |
| testcase_2_19 | All ebXML extension elements MUST have properly namespace qualified admin |
| testcase_2_20 | SOAP Envelope element must have namespace qualified admin |

| | |
|---------------|---|
| testcase_2_21 | SOAP Envelope extension element contain correct schemaLocation admin |
| testcase_2_22 | SOAP Header and Body attributes contain correct schemaLocation admin |
| testcase_2_23 | SOAP Header element contains proper namespace admin |
| testcase_2_24 | SOAP Body element contains proper namespace admin |
| testcase_2_25 | MessageHeader element must be in SOAP Header admin |
| testcase_2_26 | NON-ebXML extension elements properly namespace qualified admin |
| testcase_2_27 | An implementation of the MSH MAY ignore the namespace-qualified wildcard element admin |
| testcase_2_28 | ID attribute is assigned to ebXML MessageHeader extension element admin |
| testcase_2_28 | _1 ID attribute is assigned to each ebXML Acknowledgment extension element admin |
| testcase_2_28 | _2 ID attribute is assigned to each ebXML AckRequested extension element admin |
| testcase_2_28 | _3 ID attribute is assigned to each ebXML ErrorList extension element admin |
| testcase_2_28 | _4 ID attribute is assigned to each ebXML Manifest extension element admin |
| testcase_2_28 | _5 ID attribute is assigned to each ebXML MessageOrder extension element admin |
| testcase_2_29 | Version attribute MUST be present in ebXML MessageHeader extension element admin |
| testcase_2_29 | _1 Version attribute MUST be present in each ebXML Acknowledgment extension element admin |
| testcase_2_29 | _2 Version attribute MUST be present in each ebXML AckRequested extension element admin |
| testcase_2_29 | _3 Version attribute MUST be present in each ebXML ErrorList extension element admin |
| testcase_2_29 | _4 Version attribute MUST be present in each ebXML Manifest extension element admin |
| testcase_2_29 | _5 Version attribute MUST be present in each ebXML MessageOrder extension element admin |
| testcase_2_30 | MessageHeader version attribute is '2.0' admin |
| testcase_2_30 | _1 Acknowledgment element version attribute is '2.0' admin |
| testcase_2_30 | _2 AckRequested element version attribute is '2.0' admin |
| testcase_2_30 | _3 Each ebXML ErrorList extension element version is '2.0' admin |

| | |
|---------------|--|
| testcase_2_30 | _4 Each ebXML Manifest extension element version is '2.0' admin |
| testcase_2_30 | _5 Each ebXML MessageOrder extension element version is '2.0' admin |
| testcase_2_31 | MustUnderstand attribute set to correct namespace and value admin |
| testcase_2_32 | mustUnderstand attribute is assigned to ebXML MessageHeader extension element admin |
| testcase_2_32 | _1 mustUnderstand attribute is assigned to each ebXML Acknowledgment extension element admin |
| testcase_2_32 | _2 mustUnderstand attribute is assigned to each ebXML AckRequested extension element admin |
| testcase_2_32 | _3 mustUnderstand attribute is assigned to each ebXML ErrorList extension element admin |
| testcase_2_32 | _5 mustUnderstand attribute is assigned to each ebXML MessageOrder extension element admin |
| testcase_2_33 | _1 Not understood SOAP Header extension elements are ignored admin |
| testcase_3_1 | The MessageHeader element is REQUIRED in all ebXML Messages admin |
| testcase_3_2 | From element MUST be present in MessageHeader admin |
| testcase_3_3 | To element MUST be present in MessageHeader admin |
| testcase_3_4 | Generate error when PartyId is not a URI and type is not defined admin |
| testcase_3_5 | If type is present, it is a valid type admin |
| testcase_3_6 | If type is not present, PartyId is a valid URI admin |
| testcase_3_7 | 'Role' in From element is better defined as a URI admin |
| testcase_3_8 | 'Role' in To element is better defined as a URI admin |
| testcase_3_9 | CPAId element MUST be present in MessageHeade admin |
| testcase_3_10 | CPAId is defined as a URI admin |
| testcase_3_12 | ConversationId element present in MessageHeader admin |
| testcase_3_13 | Service element present in MessageHeader admin |
| testcase_3_14 | If the type attribute not present and Service is not a URI, then the Receiving MSH MUST report it with an errorCode of Inconsistent and severity Error admin |
| testcase_3_15 | If the type attribute is not present, the content of the Service element MUST be a URI admin |
| testcase_3_16 | Action element present in MessageHeader admin |
| testcase_3_17 | If Service not recognized, generate Error with an errorCode of |

| | |
|---------------|---|
| | NotRecognized and a severity of Error admin |
| testcase_3_17 | _1 If Action not recognized, generate Error with an errorCode of NotRecognized and a severity of Error admin |
| testcase_3_18 | MessageData element present in MessageHeader admin |
| testcase_3_19 | MessageId element present in MessageData admin |
| testcase_3_20 | Timestamp element present in MessageData admin |
| testcase_3_21 | RefToMessageId MUST be generated correctly admin |
| testcase_3_22 | Generate no RefToMessageId if first message in a conversation admin |
| testcase_3_23 | RefToMessageId element MUST be presented for Error messages admin |
| testcase_3_24 | Generate TimeToLiveExpired Error admin |
| testcase_3_25 | TimeToLive conforms to schema DateTime format admin |
| testcase_3_26 | No payload data is present in SOAP body admin |
| testcase_3_27 | xlink:href attribute is required admin |
| testcase_3_28 | xlink:role attribute SHALL have a value that is a valid URI admin |
| testcase_3_29 | location attribute in Schema element SHALL have a value that is a valid URI admin |
| testcase_3_30 | Error MUST be reported to the From Party for missing payload admin |
| testcase_3_31 | Generate Error for unresolvable CID admin |
| testcase_3_32 | Generate resolvable CID in Manifest admin |
| testcase_3_33 | Discard unreferenced payloads admin |
| testcase_4_1 | RefToMessageId element present in Error message admin |
| testcase_4_2 | Do not generate ErrorList if no errors occur admin |
| testcase_4_3 | highestSeverity attribute is required value in ErrorList element admin |
| testcase_4_4 | If any of the Error elements have a severity of Error, highestSeverity MUST set to Error admin |
| testcase_4_5 | CodeContext attribute in Error element MUST be a URI admin |
| testcase_4_6 | Namespace for codeContext is correct admin |
| testcase_4_7 | errorCode attribute present in Error message admin |
| testcase_4_8 | severity attribute present in Error element admin |
| testcase_4_9 | Generate correct severity values for Inconsistent error admin |
| testcase_4_10 | The location attribute MUST be present to point to the part of the message containing the error admin |
| testcase_4_11 | Service and Action element of an ErrorList that is included in an independent message MUST be set correctly admin |

| | |
|---------------|---|
| testcase_5_1 | if CPAId set to synchronous and SyncReply element is present, Receiving MSH resend message synchronous admin |
| testcase_5_2 | If the value of syncReplyMode in CPA is none and a SyncReply element is present, the Receiving MSH should issue an error with errorCode of Inconsistent and a severity of Error admin |
| testcase_5_3 | SyncReply must not be present if CPA syncReplyMode is set to 'none' admin |
| testcase_6_1 | If the candidate MSH fails to receive an Acknowledgment message from a receiving MSH, it should send successive retries until an Acknowledgment is received admin |
| testcase_6_2 | If the candidate MSH fails to receive an Acknowledgment message from a receiving MSH, it should send successive retries until a predetermined number of retries is exceeded admin |
| testcase_6_3 | If the sending MSH receives an Acknowledgment from a receiving MSH, it stops resending the message admin |
| testcase_6_4 | The AckRequested element must be targeted at either the Next MSH or the To Party MSH admin |
| testcase_6_5 | AckRequested element must have 'signed' attribute admin |
| testcase_6_6 | If the Sending MSH send a message with an AckRequested with the Signed attribute set to False, the Acknowledgment message must be unsigned admin |
| testcase_6_7 | If the Sending MSH send a message with an AckRequested with the Signed attribute set to true, the Acknowledgment message must be signed admin |
| testcase_6_8 | For each received message with an AckRequested with the Signed attribute set to True consistent with the CPPA, if the Receiving MSH does not support signed acknowledgment messages of the type requested, it must return Warning admin |
| testcase_6_9 | Return Error if Signature not supported and not consistent with CPA admin |
| testcase_6_10 | The Acknowledgment element generated must be targeted to the MSH node acting in the role of From Party admin |
| testcase_6_11 | An AckRequested element must not be included on a message with only an Acknowledgment element admin |
| testcase_6_12 | No AckRequested with errorCode = 'Inconsistent' and severity = 'Error' admin |
| testcase_6_13 | No AckRequested element with errorCode = 'TimeToLiveExpired' and severity = 'Error' admin |
| testcase_6_14 | If there is no SOAP actor attribute present on an Acknowledgment element, the default target is the To Party MSH admin |
| testcase_6_15 | The SOAP actor attribute of the Acknowledgment SHALL have a value corresponding to the AckRequested element of the |

| | |
|---------------|---|
| | message being acknowledged admin |
| testcase_6_16 | Timestamp must conform to a dateTime and is expressed as UTC admin |
| testcase_6_17 | a Timestamp element exists in Acknowledgment Element admin |
| testcase_6_18 | a RefToMessageId element exists in Acknowledgment Element admin |
| testcase_6_19 | RefToMessageId element contains the MessageId of the message whose delivery is being acknowledged admin |
| testcase_6_20 | Acknowledgment From PartyId value is Candidate MSH admin |
| testcase_6_21 | From PartyID of MessageHeader used if not present in Acknowledgment admin |
| testcase_6_22 | Reference element(s) present in a signed Acknowledgment admin |
| testcase_6_23 | Reference element(s) are correctly namespace qualified admin |
| testcase_6_24 | Ignore multiple Acknowledgments of same message admin |
| testcase_6_25 | No Acknowledgment element with errorCode = 'Inconsistent' and severity = 'Error' admin |
| testcase_6_26 | No Acknowledgement element with ErrorList for errorCode = 'TimeToLiveExpired' and severity = 'Error' admin |
| testcase_6_27 | If CPA have DuplicateElimination set to 'always', DuplicateElimination element must be included in receiving messages admin |
| testcase_6_28 | If CPA have DuplicateElimination set to 'never', no DuplicateElimination element must be included in receiving messages admin |
| testcase_6_29 | If CPA DuplicateElimination set to 'per message' and DuplicateElimination element is present in the sending the message, no error generated admin |
| testcase_6_30 | If CPA DuplicateElimination set to 'always' and the received message does not contain a DuplicateElimination element, the receiving MSH generate Inconsistent Error admin |
| testcase_6_31 | If CPA DuplicateElimination set to 'never' and the received message contain a DuplicateElimination element, the receiving MSH generate Inconsistent Error admin |
| testcase_6_32 | If DuplicateElimination element present and CPA DuplicateElimination set to 'always', the receiving MSH must send a message 'at-most-once' admin |
| testcase_6_33 | A Sending MSH should not resend a message with the same MessageId admin |
| testcase_6_34 | Verify Acknowledgment with PersistDuration admin |
| testcase_6_35 | Verify PersistDuration expiration admin |

| | |
|---------------|---|
| testcase_6_36 | SyncReplyMode is ignored for asynchronous communications admin |
| testcase_6_37 | Verify CPPA and SyncReply integrity admin |
| testcase_6_38 | Verify synchronous communication admin |
| testcase_6_39 | An Acknowledgement Message must be generated with RefToMessageId having the MessageId value of the message being acknowledged admin |
| testcase_6_40 | Check if Acknowledgment is returned as part of normal response admin |
| testcase_6_41 | Verify separate Acknowledgment Service name admin |
| testcase_6_42 | Verify Acknowledgment RefToMessageId value admin |
| testcase_6_43 | Verify From value of separate Acknowledgment message admin |
| testcase_6_44 | Verify To value of separate Acknowledgment message admin |
| testcase_6_45 | Verify Max Retries reached admin |
| testcase_6_46 | Verify original Acknowledgment is resent for duplicate requests admin |
| testcase_7_1 | When the MessageOrder element is present, DuplicateElimination must also be present and SyncReply must not be present admin |
| testcase_7_2 | Messages must be processed by MSH in MessageOrder admin |
| testcase_7_3 | Messages must be passed to application in correct message order admin |
| testcase_7_4 | First ordered message has a sequenceNumber of '0' admin |
| testcase_7_5 | MessageOrder status is 'Reset' for first ordered message admin |
| testcase_7_6 | SequenceNumber is reset to '0' after a Reset instruction admin |
| testcase_7_7 | For resetting SequenceNumber, status attribute must have value 'Reset' admin |
| testcase_7_8 | If SyncReply present in message, MessageOrder is NOT present admin |
| testcase_7_9 | If both MessageOrder and SyncReply are present, generate Inconsistent/Error admin |
| testcase_158 | Verify that Signature element is child of SOAP header admin |
| testcase_159 | Verify namespace of XMLDSIG admin |
| testcase_160 | Verify valid XMLDSIG structure admin |
| testcase_161 | Verify first Signature signs message admin |
| testcase_162 | Signature required for entire message admin |
| testcase_163 | Signature is rendered according to XMLDSIG specification admin |

| | |
|---------------|---|
| testcase_164 | SignedInfo has CanonicalizationMethod, SignatureMethod and one or more Reference elements admin |
| testcase_164_ | 1 SignatureMethod Algorithm attribute is present admin |
| testcase_165 | SignatureMethod Algorithm attribute is present admin |
| testcase_166 | SignedInfo has CanonicalizationMethod with value of 'http://www.w3.org/TR/2001/REC-xml-c14n-20010315' admin |
| testcase_167 | SignedInfo has CanonicalizationMethod with Algorith value of 'http://www.w3.org/2000/09/xmldsig#dsa-sha1' admin |
| testcase_168 | Signature is validated and message is passed to the application admin |
| testcase_169 | Verify Type attribute of Reference admin |
| testcase_170 | Verify Transform sub-element of Reference admin |
| testcase_171 | Generate Transform XPath element excluding SOAP nextMSH or next SOAP node admin |
| testcase_172 | Verify last Transform Algorithm attribute value admin |
| testcase_173 | Verify URI of Signature points to message payloa admin |
| testcase_174 | Verify URI of Signature points to message payload admin |
| testcase_175 | Verify URI of Signature points to message payload admin |
| testcase_176 | Digitally signed inbound message gets a digitally signed Acknowledgment admin |
| testcase_179 | Signing takes place prior to encryption admin |

NDR 2.0

| Rule ID | Requirements |
|----------------|---|
| Rule 007 | Use of Lower-Camel-Case in attribute name |
| Rule 008 | Use of Upper-Camel-Case in element and type name |
| Rule 010 | Use of alphabet in element, attribute and type name |
| Rule 011 | No use of charaters not allowed |
| Rule 012 | Use of allowed acronyms, abbreviations, or other word truncations |
| Rule 038 | targetNamespace declaration |
| Rule 041 | Namespace definition as URN |
| Rule 042 | Namespace in draft status |
| Rule 043 | Namespace in specification status |
| Rule 046 | Schema location declaration |
| Rule 048 | Use of schema version attribute |
| Rule 049 | Use of version attribute template |
| Rule 056 | elementFormDefault declaration |
| Rule 057 | attributeFormDefault declaration |
| Rule 059 | No applInfo |
| Rule 060 | No notation |
| Rule 061 | No wildcard |

| | |
|----------|--|
| Rule 062 | No any |
| Rule 063 | No anyAttribute |
| Rule 065 | No substitutionGroup |
| Rule 073 | No nillable attribute |
| Rule 074 | No empty element |
| Rule 076 | No all element |
| Rule 080 | xsd:type in restriction |
| Rule 082 | Use of token rsm (Root Schema) |
| Rule 083 | Import of RABIE, UDT, QDT modules (Root Schema) |
| Rule 087 | Root element declaration (Root Schema) |
| Rule 088 | No separators and spaces in root element name (Root Schema) |
| Rule 090 | one xsd:complexType (Root Schema) |
| Rule 092 | Root element annotation (Root Schema) |
| Rule 032 | Use of ram:ReusableAggregateBusinessInformationEntity schema module (RABIE) |
| Rule 079 | RABIE no xsd:extension (RABIE) |
| Rule 096 | Import of UDT, QDT modules (RABIE) |
| Rule 098 | complexType name (RABIE) |
| Rule 099 | Use of xsd:sequence and/or xsd:choice (RABIE) |
| Rule 100 | No recursion of sequence and/or choice (RABIE) |
| Rule 105 | element declaration (RABIE) |
| Rule 026 | Use of cct:CoreComponentType schema module (CCT) |
| Rule 079 | CCT xsd:type in extension (CCT) |
| Rule 117 | The use of token cct (CCT) |
| Rule 118 | No import or include (CCT) |
| Rule 119 | Use of a named xsd:complexType (CCT) |
| Rule 120 | complexType name (CCT) |
| Rule 121 | one xsd:simpleContent in xsd:complexType (CCT) |
| Rule 122 | one xsd:extension element in xsd:complexType definition xsd:simpleContent element (CCT) |
| Rule 129 | xsd:complexType annotation (CCT) |
| Rule 130 | xsd:attribute annotation (CCT) |
| Rule 028 | Use of udt:UnqualifiedDataType schema module (UDT) |
| Rule 079 | UDT xsd:type in extension (UDT) |
| Rule 131 | The use of token udt (UDT) |
| Rule 132 | Import of identifierList codeList modules (UDT) |
| Rule 134 | UDT name (UDT) |
| Rule 136 | one xsd:restriction in xsd:simpleType (UDT) |
| Rule 138 | one xsd:simpleContent in xsd:complexType (UDT) |
| Rule 139 | one xsd:extension element in xsd:complexType definition xsd:simpleContent element (UDT) |
| Rule 148 | xsd:complexType or xsd:simpleType annotation (UDT) |
| Rule 149 | xsd:attribute annotation (UDT) |
| Rule 030 | Use of qdt:QualifiedDataType schema module (QDT) |
| Rule 079 | QDT no xsd:extension (QDT) |
| Rule 150 | The use of token qdt (QDT) |
| Rule 151 | Import of UDT modules (QDT) |
| Rule 160 | one xsd:restriction in xsd:complexType definition xsd:simpleContent element (QDT) |
| Rule 161 | xsd:complexType or xsd:simpleType annotation (QDT) |

| | |
|----------|---|
| Rule 162 | xsd:attribute annotation (QDT) |
| Rule 079 | Code List no xsd:extension (Code Lists) |
| Rule 169 | schemaLocation declaration as URL (Code Lists) |
| Rule 171 | No import (Code Lists) |
| Rule 172 | One simpleType (Code Lists) |
| Rule 173 | xsd:simpleType name (Code Lists) |
| Rule 174 | Restriction element base attribute value set to xsd:token (Code Lists) |
| Rule 175 | xsd:value attribute in xsd:enumeration element (Code Lists) |
| Rule 179 | xsd:enumeration annotation (Code Lists) |
| Rule 079 | Identifier List Schema no xsd:extension (Identifier List) |
| Rule 186 | schemaLocation declaration as URL (Identifier List) |
| Rule 188 | No import or include (Identifier List) |
| Rule 189 | One simpleType (Identifier List) |
| Rule 190 | xsd:simpleType name (Identifier List) |
| Rule 191 | Restriction element base attribute value set to xsd:token (Identifier List) |
| Rule 192 | xsd:value attribute in xsd:enumeration element (Identifier List) |
| Rule 197 | xsd:enumeration annotation (Identifier List) |

CCTS

| Rule ID | Requirements |
|-----------|--|
| Rule B001 | BBIE, ABIE, or ASBIE |
| Rule B006 | Aggregate Business Information Entity including property |
| Rule B019 | No use of consecutive redundant words |
| Rule B021 | No non-alphanumeric characters |
| Rule B022 | No use of and, of, the |
| Rule B024 | the Dictionary Entry Name of a BBIE |
| Rule B025 | the Dictionary Entry Name of a ASBIE |
| Rule B026 | Components of a Dictionary Entry Name |
| Rule B030 | Dictionary Entry Name of a ABIE |
| Rule C007 | Use of approved Core Component Type |
| Rule C030 | Dictionary Entry Name of a Core Component Type |
| Rule C031 | the name of the Representation Term |
| Rule D010 | No use of consecutive redundant words |
| Rule D011 | No use of non-alphanumeric characters |
| Rule D012 | No use of and, of, the |
| Rule D014 | Dictionary Entry Name of a Data Type |
| Rule D015 | the name of the Representation Term |